GEORGIA INSTITUTE OF TECHNOLOGY
Engineering Experiment Station
Atlanta, Georgia

FINAL REPORT

PROJECT A-831

STUDY OF THE METHODS FOR THE NUMERICAL SOLUTION OF ORDINARY DIFFERENTIAL EQUATIONS

By

L.J. GALLAHER, G.E. DUNCAN, O.B. FRANCIS, JR.,
J.M. GWYNN, JR., H.G. HALE, JR., and I.E. PERLIN

CONTRACT NAS8-20014

26 JANUARY 1966 to 25 JANUARY 1967

Performed for
GEORGE C. MARSHALL SPACE FLIGHT CENTER
NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
HUNTSVILLE, ALABAMA

132

ABSTRACT

This report outlines the work accomplished and results achieved in the preparation of a computer procedure for the integration of ordinary differential equations. The following characteristics of the programs are specified:

a) The procedure for integration must achieve the accuracy specified by the user.

b) The procedure must be problem independent and applicable to the integration of any degree or number of coupled differential equations.

c) The step size, order and method of integration are to be chosen so as to minimize computation time while meeting the accuracy requirements.

d) The procedure is to have built-in learning so that previous experience can be used from one call to the next to decide on the method and order to be used. The procedure is to be self-modifying.

The following methods were used in the development of the procedure.

a) Adams-Bashforth-Moulton

b) Stetter-Gragg-Butcher

c) Cowell's method of constant Nth order difference

d) Runge-Kutta-Shanks.

Four different orders were used for each of the above methods.

Information is provided on an executive procedure developed to act in an administrative and bookkeeping capacity for the basic integration routines indicated above, plus a start and restart routine, which contains a separate Runge-Kutta-Shanks routine. This executive procedure works very satisfactorily.

Three types of problems were used to exercise this procedure. These three types are the Arenstorf orbits of the restricted three body problem, the system of linear differential equations associated with Fourier transforms, and the system of linear equations obtained from the partial differential equation for the vibrating string.

The results of running with a variety of problems and accuracies are that no particular method seems very superior to any other. All methods performed well.

The results justify the conclusion that the program developed would be very useful as a general library program for integrating systems of differential equations.

Several suggestions for further study are outlined in Chapter IV.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF TABLES

I. INTRODUCTION

In previous work done under this contract, an effort was made to determine which of the many methods and orders available for integrating ordinary differential equations was best. While it was possible to show that, under certain circumstances, some methods and orders outperformed others, no one method was clearly superior under all circumstances.

In the present contract, the goal was set of writing a computer program for the integration of systems of ordinary differential equations (initial value problems), characterized by the following specifications:

a) The integration must meet a (user's) specified accuracy.

b) The procedure will be problem independent and applicable to the integration of any degree or number of coupled differential equations.

c) The step size, order, and method of integration are to be chosen by the procedure so as to be optimum; that is, to minimize the computation time while meeting the accuracy requirements.

d) The procedure will have built-in learning so that it can use its experience from one call to the next to decide on the method and order to be used. The procedure will be self-modifying.

The methods used are as follows:

(1) The Adams-Bashforth-Moulton method,

(2) The Stetter-Gragg-Butcher method,

(3) Cowell's method of constant Nth order differences,

(4) The Runge-Kutta-Shanks method.

With each of these methods, four different orders are used. A history file is kept showing the past performance scores of each method and order and is used to select which methods and orders are to be employed.

The program works in the following way. When a call is made in the procedure to integrate from point a to the point b, this interval is divided into eighths. The first eighth of the interval is integrated by one method for each of two different orders, and the times taken by each recorded. The second eighth is integrated by another method, also for two different orders, and the times recorded. The winners then compete against each other over the next fourth of the interval. That is, the fastest order of the first method and the faster order of the second method are both used to integrate the second fourth of the interval, and the times taken by each recorded. The faster method of these two is then presumably the best (fastest) of the four tried, and it is used (alone) to integrate over the last half of the interval. All of the times measured above are then logged in a cumulative history file and the winners and losers noted.

This history file then is used as the basis for selecting which methods and orders are chosen each time.

The first of the two methods is chosen at random (using a random number generator) from among the four available. The second method is chosen to be the method showing the best history of success among the three remaining methods, with the cumulative history file being used to determine the degree of success. Then within each method the same kind of selection process with respect to orders is used. That is, the first order is chosen at random, and the second order is chosen on the basis of which of the remaining three has been the most successful (fastest running) order of that method. Thus it is seen that the past performance of the different methods and orders influences the choice of which are allowed to compete, such that the more successful have a higher probability of being selected.

In using time as the sole estimate of performance efficiency, it is assumed that all orders and methods have satisfactorily met the accuracy requirements. The accuracy requirements of each method are met by controlling step size and making error estimates at each step. The method of error estimate is different for the different methods. In the Runge-Kutta single step method, the error is estimated by taking two half steps and then a whole step. In the Adams and Butcher methods the difference between predictor and corrector is used. In the Cowell method a mid-range formula is used. (Only in the Adams and the Runge-Kutta cases is there good theoretical justification for using these methods to calculate the actual error -- the error estimates in the Butcher and Cowell methods are essentially empirical.)

One further feature introduced into the learning process is the gradual "forgetting" of events in the more distant past. This causes the events in the distant past to have less influence than those more recent in determining the score or performance figure of an order and method.

Three types of problems were used to exercise the integration procedure:

First, the Arenstorf type orbits of the restricted three body problem (four equations).

Second, the system of linear differential equations associated with the Fourier transforms (20 to 40 equations).

Third, the system of linear differential equations obtained from a discretization of the partial differential equation for the vibrating string (50 to 100 equations).

The first of these is characterized by the necessity of frequent step size change. The other two have no need for step size change once the correct step is found.

The preliminary results of running with a variety of problems and accuracies is that no particular method seems to be exceptionally superior to any other. It did appear that, for the accuracy range used, certain orders of some methods were inappropriate. Also, for a given method one particular order usually dominated, but which one dominated depended on the accuracy being asked and to some extent the problem. In any event the program adapted quite rapidly to the characteristics of a particular problem and accuracy.

All methods performed well and, for different problems, different methods showed up more successfully. For example, the Runge-Kutta method was most successful when frequent step size changes were required, but the multistep methods performed better when long runs of uniform step size were appropriate.

The results justify the conclusion that the present program would be suitable and effective as a general library program for integrating systems of differential equations.

II. INTEGRATION METHODS

A. The Method of Adams, Bashforth and Moulton

1. Description of the Method

The method investigated consists of the combination of two different versions of the method of Adams into a predictor-corrector system [5]. The use of this system to obtain numerical solutions to a set of simultaneous differential equations with given initial conditions is independent both of the number of equations in the set to be solved and of the orders of the individual equations in the set; provided, however, that each equation of order m is expressed as a set of m coupled first order equations.

In general then, one deals with the system of equations

$$\vec{y}\,'(x) \equiv \frac{d}{dx}\,\vec{y}(x) = \vec{f}\Big(x,\vec{y}(x)\Big), \qquad (1\text{-}1)$$

where $\vec{y}\,'$, $\vec{y}$, and $\vec{f}$ are vectors, each having a number of components, N, equal to $\sum_{i=1}^{k} m_i$, where k is the number of equations in the set to be solved, and the m_i are their individual orders.

This vector differential equation is equivalent to the integral equation

$$\vec{y}(x+h) = \vec{y}(x) + \int_x^{x+h} \vec{f}\Big(t,\vec{y}(t)\Big)dt. \qquad (1\text{-}2)$$

At the point $x = x_q \equiv x_{q-1} + h$, this integral is approximated first by

$$\vec{y}_q^{\,(0)} = \vec{y}_{q-1} + h\sum_{\mu=0}^{q-1} \beta_{q-1,q-1-\mu}\vec{f}_\mu \qquad (1\text{-}3a)$$

and then repeatedly by

$$\vec{y}_q^{(\nu+1)} = h\beta^*_{q,o}\vec{f}\left(x_q,\vec{y}^{(\nu)}(x_q)\right) + h\sum_{\mu=0}^{q-1} \beta^*_{q,q-\mu}\, \vec{f}_\mu$$

$$\equiv h\beta^*_{q,o}\vec{f}_q^{(\nu)} + \vec{C}, \quad \nu=0,1,2,\ . \ . \ . \qquad (1\text{-}3b)$$

which converges toward $\vec{y}_q \equiv \vec{y}(x_q)$ as ν increases. Formula (1-3a) is called the Adams-Bashforth predictor equation, and formula (1-3b) is the Adams-Moulton corrector.

The coefficients $\beta_{q\rho}$ and $\beta^*_{q\rho}$ are derived by the equivalent of integrating Lagrangian polynomials fitted to $\vec{f}$, but are independent of both $\vec{f}$ and h. The polynomial for the predictor is of degree q-1 passing through the q points $\vec{f}_0, \vec{f}_1, \ . \ . \ ., \vec{f}_{q-1}$, while that for the corrector is of degree q passing through the q + 1 points $\vec{f}_0, \vec{f}_1, \ . \ . \ ., \vec{f}_q$.

An explicit formula for the $\beta_{q\rho}$ is

$$\beta_{q\rho} = (-1)^\rho\left\{\binom{\rho}{\rho}\gamma_\rho + \binom{\rho+1}{\rho}\gamma_{\rho+1} + \ . \ . \ . + \binom{q}{\rho}\gamma_q\right\}, \quad \begin{matrix} q = 0,1,2,\ . \ . \ . \\ \rho = 0,1,\ . \ . \ ., q \end{matrix}$$

where the $\binom{\rho+i}{\rho}$ represent binomial coefficients and the γ_ρ are found by the recursion relation

$$\gamma_\rho + \tfrac{1}{2}\gamma_{\rho-1} + \ . \ . \ . + \frac{1}{\rho+1}\gamma_o = 1, \qquad \rho = 0,1,2,\ . \ . \ ., \quad .$$

and an explicit formula for the $\beta^*_{q\rho}$ is

$$\beta^*_{q\rho} = (-1)^\rho\left\{\binom{\rho}{\rho}\gamma^*_\rho + \binom{\rho+1}{\rho}\gamma^*_{\rho+1} + \ . \ . \ . + \binom{q}{\rho}\gamma^*_q\right\}, \quad \begin{matrix} q = 0,1,2,\ . \ . \ . \\ \rho = 0,1,\ . \ . \ ., q \end{matrix}$$

where $\gamma^*_o = 1$ and $\gamma^*_\rho = \gamma_\rho - \gamma_{\rho-1}$, $\rho = 1,2,3,\ . \ . \ . \ .$

Bounds on the errors for the two approximations are the maximums within the interval $[x_o,x_q]$ of

$$\left| \gamma_q h^{q+1} \frac{d^{q+1}}{dx^{q+1}} \vec{y} \right| \qquad \text{(for Adams-Bashforth)} \qquad (1\text{-}4a)$$

and of

$$\left| \gamma^*_{q+1} h^{q+2} \frac{d^{q+2}}{dx^{q+2}} \vec{y} \right| \qquad \text{(for Adams-Moulton)} \qquad (1\text{-}4b)$$

and M, the order of the predictor-corrector system, is assumed to approximate that of the corrector, which is q + 1.

2. The Computer Procedure

The procedure ADAMS itself is written to be included in programs written in single precision for the Burroughs B-5500 computer. The language is Algol 60 augmented by additional features available in the Algol compiler for the B-5500. There are no unusual hardware requirements, because all input and output to the procedure is under control of the including program through the formal parameter list. All variables not in the formal parameter list are local to the procedure, and no files are used by the procedure.

2.1 Parameters and Variables

The following lists of formal parameters and local variables will be useful in describing the operation of procedure ADAMS. In the remainder of this discussion the interchange of upper and lower case letters, necessitated by approximating the notation [5] within the limited character set available to a computer; is straight-forward and will be done freely without further comment.

Formal Parameters

Identifier	Type	Usage or Meaning
N	Integer	The number of components in the vectors $\vec{y}$, $\overrightarrow{EA}$, and $\overrightarrow{ER}$.
XI	Real	Initial value of the independent variable.
XF	Real	Final value of the independent variable.
Y	Real Array	Current dependent variable vector. Contains initial values at entry and final values at exit.
F	Procedure	Calculates the vector $\vec{f}\left(x,\vec{y}(x)\right)$.
P	Real	Power of C1 used in error control.
Q	Integer	Number of back $\vec{f}$ points used in the approximating polynomials. One less than M, the order of the method.
DX	Real	Upper bound on the initial step size.
EA	Real Array	Absolute error bound vector.
ER	Real Array	Relative error bound vector.
ADAMSCOEFF	Real Array	Contains the $\beta_{q\rho}$, the $\beta^*_{q\rho}$, and $\left|1 - \gamma_{q+1}/\gamma^*_{q+1}\right|$.
RKSFNS	Integer	Function evaluations per step for procedures START and SHANKS.
RKSORDER	Integer	Order of R.K.S. method to be used by START and SHANKS.
RKSCOEFF	Real Array	Coefficients for START and SHANKS. See the desciptions of START and SHANKS elsewhere in this report for details.
START	Integer Procedure	Gives the necessary points for starting and restarting. Name contains the factor by which C1 is multiplied to coordinate step size between START and ADAMS.
SHANKS	Procedure	Used to complete fractional steps at the ends of intervals.

The procedures START, SHANKS, and F, as well as the coefficient arrays ADAMSCOEFF and RKSCOEFF, are not a part of the procedure ADAMS and must be included separately in all programs using ADAMS (see 2.2 and 2.3).

Local Variables

Identifier	Type	Usage or Meaning
X	Real	x_q, current value of the independent variable.
INTERVAL	Real	XF - XI, the interval of integration.
C1	Integer	Two to an integral power. Determines H.
H	Real	INTERVAL / C1, the current step size.
C2	Real	Number of steps of size H remaining from X to XF.
GR	Real	$\left\| 1 - \gamma_{q+1}/\gamma^*_{q+1} \right\|$, used with CHANGE and ERROR.
CHANGE	Real	Controls the number of iterations of the corrector equation.
ERROR	Real	Controls the error and running time through the step size.
FP	Real Array	Predicted $\vec{y}'_q$ vector, $\vec{f}^{(p)}$, the $\vec{f}_q^{(\nu)}$ of (1-3b).
FC	Real Array	Corrected $\vec{y}'_q$ vector, $\vec{f}^{(c)}$, $\vec{f}_q^{(\nu+1)}$ in (1-3b).
FH	Real Array	$\vec{f}$ history vector. Contains 2q-1 back points for each of the N components of f.
YP	Real Array	Predicted $\vec{y}_q$ vector, $\vec{y}^{(p)}$, the $\vec{y}_q^{(o)}$ of (1-3a).
YC	Real Array	Corrected $\vec{y}_q$ vector, $\vec{y}^{(c)}$, the $\vec{y}_q^{(\nu+1)}$ of (1-3b).
YB	Real Array	Back $\vec{y}$ vector, $\vec{y}^{(b)}$, needed for restarting after halving.
YD	Real Array	Alternate YB.

All local arrays are dynamic with respect to N and Q and, to avoid moving large numbers of components, reversals in meaning are made on successive steps or iterations between FP and FC, between YP and YC, and between YB and YD. The FH vector array is indexed cyclically for the same reason. For further details consult the flow diagram and the listing of procedure ADAMS following this discussion.

2.2 The F Procedure

A procedure for calculating the vector $\vec{y}\,' = \vec{f}\left(x,\vec{y}(x)\right)$ must be included global to a call for procedure ADAMS for each set of differential equations to be solved by a program using ADAMS. This procedure is called by ADAMS as the formal parameter F and must itself have the following formal parameter list:

Identifier	Type	Usage or Meaning
N	Integer	Number of components in the vectors YV and FV.
X	Real	Current value of the independent variable.
YV	Real Array	Current dependent variable vector (input).
FV	Real Array	F value vector (output).

N and X may be called by value. The arrays YV and FV are one-dimensional starting at zero and must be called by name.

2.3 Orders Available

The procedure ADAMS is written to be completely general with regard to order, and any order may be used if the necessary coefficients are placed in the ADAMSCOEFF array. For a given order M = q + 1, there are 2q + 2 = 2M coefficients which should appear in the array beginning at position zero in the following order:

$$\beta_{q-1,q-1}, \beta_{q-1,q-2}, \ldots, \beta_{q-1,o}, \beta^{*}_{q,q}, \beta^{*}_{q,q-1}, \ldots, \beta^{*}_{q,o}, \left|1-\gamma_{q+1}/\gamma^{*}_{q+1}\right|.$$

2.4 Starting an Integration

Since the Adams method is a multistep method it cannot start itself but must rely on a starting procedure that will supply at least q-1 $\vec{f}$ points which, together with a given initial $\vec{f}$ point and a current $\vec{y}$ point, comprise a history upon which it can build. The starting procedure used here is the Runge-Kutta-Shanks procedure START, described elsewhere in this report. The number of function evaluations per step and the order of Runge-Kutta-Shanks method used by START may be varied at will by the user through the formal parameters of ADAMS. This will achieve optimum compatibility with the order of Adams method being used for each given set of differential equations being solved.

Initial step size is determined by the formal parameter DX. The initial trial start will be made with a step H = INTERVAL / C1, where C1 is set to the smallest integer power of two such that $|H| \le |DX|$ and $|H| \le |\text{INTERVAL}|/Q$. This causes the prodecure ADAMS to take at least one step after starting regardless of the magnitude of DX. If the procedure START cannot meet the error requirements at the initial H, it doubles C1 repeatedly until these requirements can be met.

2.5 Error Estimates and Step Size Control

To minimize running time without introducing errors intolerably large, the error in each component of the final $\vec{Y}$ vector is controlled through the use of the formal parameters $\vec{EA}$ and $\vec{ER}$. $\vec{EA}$ specifies the maximum allowable absolute magnitude of the error in each component of $\vec{Y}$, and $\vec{ER}$ specifies the maximum allowable relative magnitude. These two error control vectors are used in conjunction with the quantity GR = $|1-\gamma_q/\gamma^*_{q+1}|$, which is derived from the bounds (1-4), and a parameter P, chosen from the interval $[\frac{1}{2},1]$ by empirical determination of the randomness of the round-off error in a particular set of differential equations. ($P = \frac{1}{2}$ corresponds to totally random error and $P = 1$ corresponds to totally additive error.) In practice γ_{q+1} has been used in GR instead of γ_q to be conservative, because the quantity being controlled is only an estimate of the true error.

The estimated error vector $\overrightarrow{\text{ERROR}}$ is defined to be $|\vec{y}^{(c)} - \vec{y}^{(p)}|$, where $\vec{y}^{(p)}$ is the $\vec{y}_q^{\,(o)}$ of (1-3a) and $\vec{y}^{(c)}$ is $\vec{y}_q^{\,(\nu_f+1)}$ in (1-3b), with ν_f being the first ν for which every component of

$$\overrightarrow{\text{CHANGE}} \equiv |\vec{f}_q^{\,(\nu+1)} - \vec{f}_q^{\,(\nu)}| = \left|\left(\vec{y}_q^{\,(\nu+1)} - \vec{y}_q^{\,(\nu)}\right) / h\beta^*_{q,o}\right|$$

is less than the corresponding component of either

$$\left|\frac{\vec{EA} \cdot GR}{C1^P \cdot 2^{Q+5} \cdot h\beta^*_{q,o}}\right| \quad \text{or} \quad \left|\frac{\vec{ER} \cdot GR}{C1^P \cdot 2^{Q+5} \cdot h\beta^*_{q,o}} \cdot \vec{f}_q^{\,(\nu+1)}\right| .$$

If any component of $\overrightarrow{\text{ERROR}}$ is larger than the corresponding components of both

$$\left|\frac{\vec{EA} \cdot GR}{C1^P}\right|$$

and

$$\left|\frac{\vec{ER} \cdot GR}{C1^P} \cdot \vec{y}^{(c)}\right| ,$$

then $\vec{y}$ is replaced by $\vec{y}^{(b)}$, the step size is halved, and q-1 new $\vec{f}$ points and a new current $\vec{y}$ are obtained from the procedure START. If it is not necessary to halve the step size, then $\vec{y}^{(c)}$ becomes the new $\vec{y}$. If every component of $\overrightarrow{\text{ERROR}}$ is smaller for three consecutive steps than the corresponding components of both

$$\left|\frac{\vec{EA}\cdot GR}{C1^{P}\cdot 2^{Q+5}}\right| \text{ and } \left|\frac{\vec{ER}\cdot GR}{C1^{P}\cdot 2^{Q+5}}\cdot \vec{y}^{(c)}\right| ,$$

then if there are at least 2q-1 back points in the $\vec{FH}$ array and there are at least two more steps of the current size necessary to reach XF, the step size is doubled before the next trial step. If it is not necessary either to halve or to double the step size, X is increased by H and a new trial step is made.

2.6 Finishing an Integration

The procedure ADAMS continues as described until XF is reached unless repeated halvings and doublings of the step size bring the independent variable to within a fraction of a single step of XF. When this occurs, the fractional step is completed by the Runge-Kutta-Shanks procedure SHANKS, described elsewhere in this report. The order of Runge-Kutta-Shanks method and the number of function evaluations per step used here will be the same for a given integration as those used by the procedure START.

3. Flow Diagram and Program Listing

Figure 1 is the flow diagram for the method of Adams, Bashforth and Moulton. The program listing follows at the end of this section.

4. Results and Conclusions

For experimental and diagnostic reasons, the procedure ADAMS was originally checked out with separate arrays for the dependent variable vectors $\vec{YI}$ (initial values), $\vec{YB}$ (back values), and $\vec{YF}$ (final values), all of which are now made equivalent to $\vec{Y}$ in the DEFINE statement. These arrays may be removed from the define statement and declared to be local arrays, global arrays, or formal parameters if any reason for doing so should arise. If this is done $\vec{Y}$ should be removed from the formal parameter list.

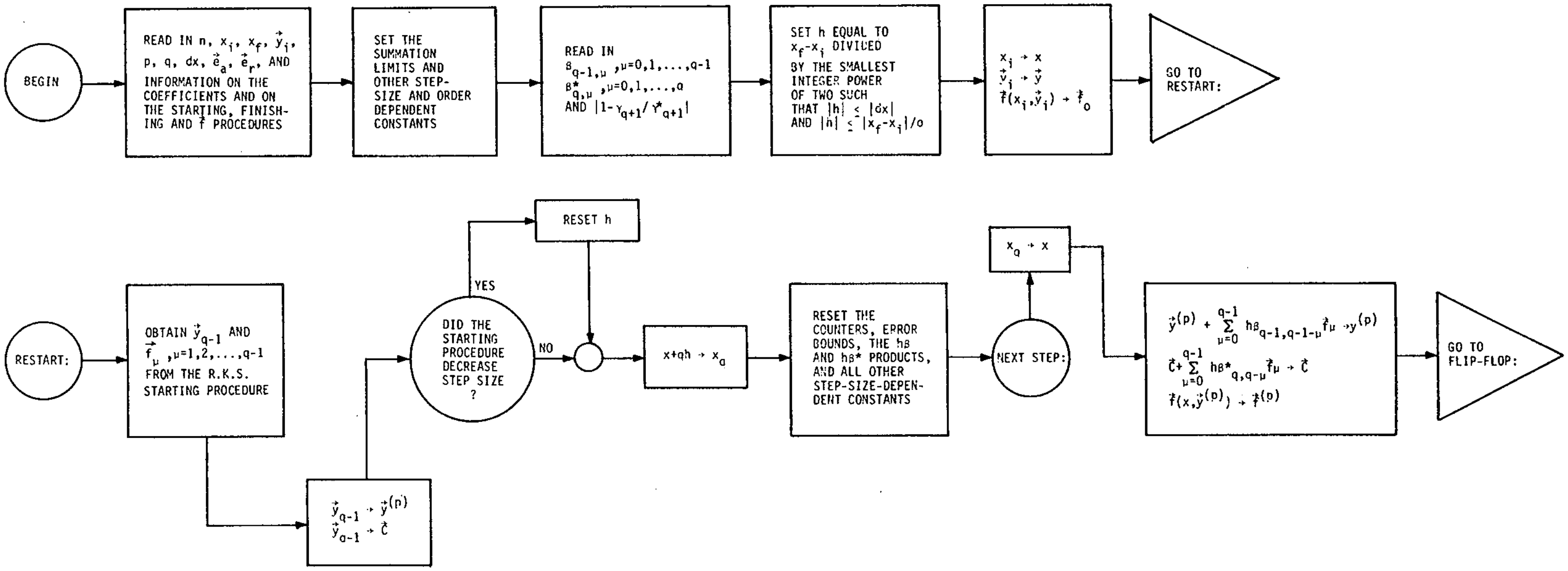
BEGIN
READ IN n, x_i, x_f, $\vec{y}_i$, p, q, dx, $\vec{e}_a$, $\vec{e}_r$, AND INFORMATION ON THE COEFFICIENTS AND ON THE STARTING, FINISHING AND $\vec{f}$ PROCEDURES
SET THE SUMMATION LIMITS AND OTHER STEP-SIZE AND ORDER DEPENDENT CONSTANTS
READ IN $\beta_{q-1,\mu}$,μ=0,1,...,q-1 $\beta^*_{q,\mu}$,μ=0,1,...,q AND $|1-\gamma_{q+1}/\gamma^*_{q+1}|$
SET h EQUAL TO x_f-x_i DIVIDED BY THE SMALLEST INTEGER POWER OF TWO SUCH THAT $|h| \leq |dx|$ AND $|h| \leq |x_f-x_i|/o$
$x_i \to x$
$\vec{y}_i \to \vec{y}$
$\vec{f}(x_i,\vec{y}_i) \to \vec{f}_o$
GO TO RESTART:
RESET h
RESTART:
OBTAIN $\vec{y}_{q-1}$ AND $\vec{f}_\mu$,μ=1,2,...,q-1 FROM THE R.K.S. STARTING PROCEDURE
YES
DID THE STARTING PROCEDURE DECREASE STEP SIZE ?
NO
$x+qh \to x_q$
RESET THE COUNTERS, ERROR BOUNDS, THE $h\beta$ AND $h\beta^*$ PRODUCTS, AND ALL OTHER STEP-SIZE-DEPENDENT CONSTANTS
$x_q \to x$
NEXT STEP:
$\vec{y}^{(p)} + \sum_{\mu=0}^{q-1} h\beta_{q-1,q-1-\mu}\vec{f}_\mu \to y^{(p)}$
$\vec{C}+\sum_{\mu=0}^{q-1} h\beta^*_{q,q-\mu}\vec{f}_\mu \to \vec{C}$
$\vec{f}(x,\vec{y}^{(p)}) \to \vec{f}^{(p)}$
GO TO FLIP-FLOP:
$\vec{y}_{q-1} \to \vec{y}^{(p)}$
$\vec{y}_{q-1} \to \vec{C}$

Figure 1. Flow Diagram for the Adams Method.

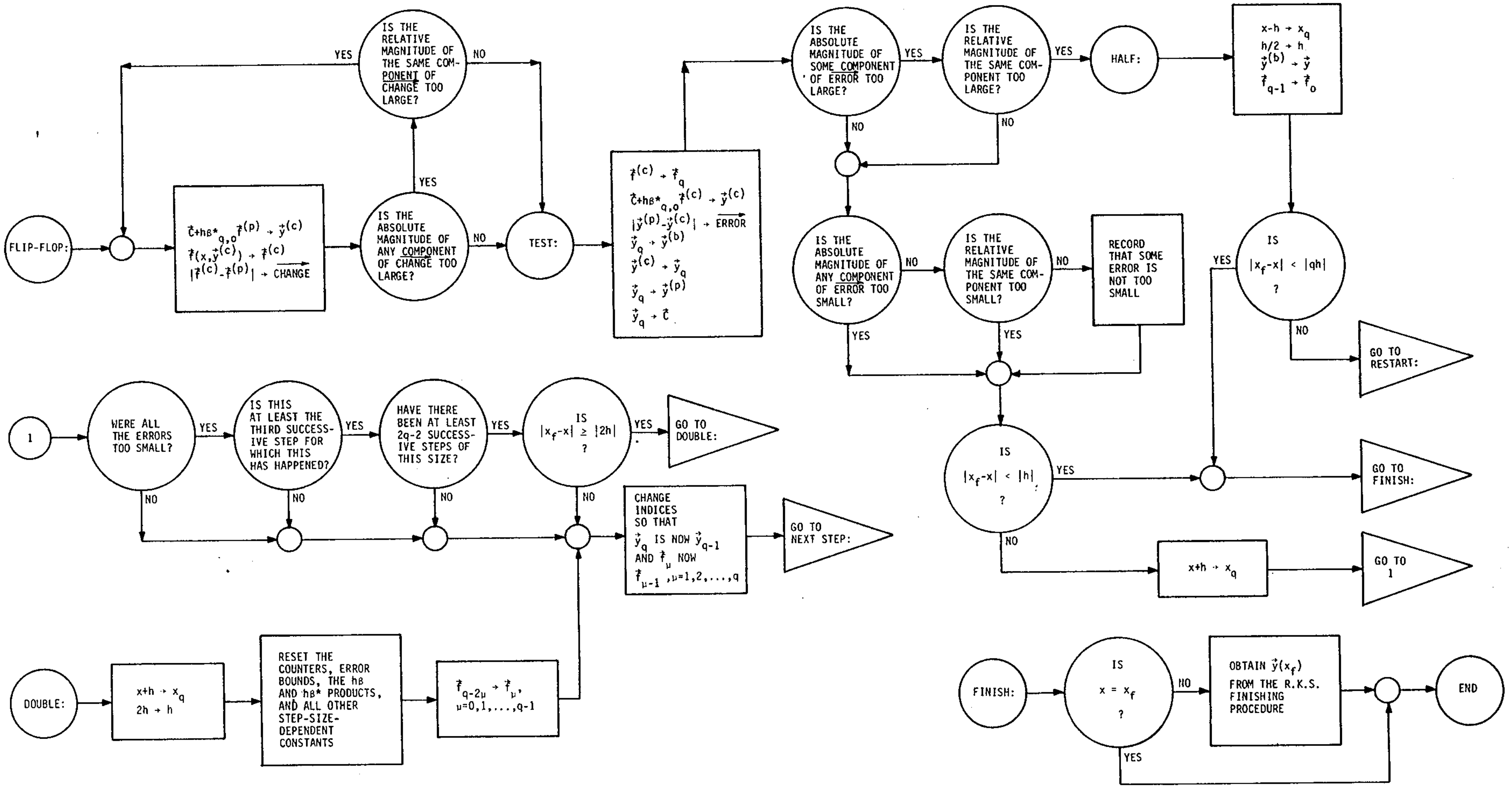
FLIP-FLOP:
$\vec{c}+h\beta^*_{q,o}\vec{f}^{(p)} \to \vec{y}^{(c)}$
$\vec{f}(x,\vec{y}^{(c)}) \to \vec{f}^{(c)}$
$|\vec{f}^{(c)}-\vec{f}^{(p)}| \to \overrightarrow{\text{CHANGE}}$
IS THE ABSOLUTE MAGNITUDE OF ANY COMPONENT OF CHANGE TOO LARGE?
YES
NO
IS THE RELATIVE MAGNITUDE OF THE SAME COMPONENT OF CHANGE TOO LARGE?
YES
NO
TEST:
$\vec{f}^{(c)} \to \vec{f}_q$
$\vec{c}+h\beta^*_{q,o}\vec{f}^{(c)} \to \vec{y}^{(c)}$
$|\vec{y}^{(p)}-\vec{y}^{(c)}| \to \overrightarrow{\text{ERROR}}$
$\vec{y}_q \to \vec{y}^{(b)}$
$\vec{y}^{(c)} \to \vec{y}_q$
$\vec{y}_q \to \vec{y}^{(p)}$
$\vec{y}_q \to \vec{c}$
IS THE ABSOLUTE MAGNITUDE OF SOME COMPONENT OF ERROR TOO LARGE?
YES
NO
IS THE RELATIVE MAGNITUDE OF THE SAME COMPONENT TOO LARGE?
YES
NO
HALF:
$x-h \to x_q$
$h/2 \to h$
$\vec{y}^{(b)} \to \vec{y}$
$\vec{f}_{q-1} \to \vec{f}_o$
IS THE ABSOLUTE MAGNITUDE OF ANY COMPONENT OF ERROR TOO SMALL?
NO
YES
IS THE RELATIVE MAGNITUDE OF THE SAME COMPONENT TOO SMALL?
NO
YES
RECORD THAT SOME ERROR IS NOT TOO SMALL
IS $|x_f-x| < |qh|$?
YES
NO
GO TO RESTART:
IS $|x_f-x| < |h|$?
YES
NO
GO TO FINISH:
$x+h \to x_q$
GO TO 1
1
WERE ALL THE ERRORS TOO SMALL?
YES
NO
IS THIS AT LEAST THE THIRD SUCCESSIVE STEP FOR WHICH THIS HAS HAPPENED?
YES
NO
HAVE THERE BEEN AT LEAST 2q-2 SUCCESSIVE STEPS OF THIS SIZE?
YES
NO
IS $|x_f-x| \geq |2h|$?
YES
NO
GO TO DOUBLE:
CHANGE INDICES SO THAT $\vec{y}_q$ IS NOW $\vec{y}_{q-1}$ AND $\vec{f}_\mu$ NOW $\vec{f}_{\mu-1}$, $\mu=1,2,\ldots,q$
GO TO NEXT STEP:
DOUBLE:
$x+h \to x_q$
$2h \to h$
RESET THE COUNTERS, ERROR BOUNDS, THE $h\beta$ AND $h\beta^*$ PRODUCTS, AND ALL OTHER STEP-SIZE-DEPENDENT CONSTANTS
$\vec{f}_{q-2\mu} \to \vec{f}_\mu$, $\mu=0,1,\ldots,q-1$
FINISH:
IS $x = x_f$?
NO
YES
OBTAIN $\vec{y}(x_f)$ FROM THE R.K.S. FINISHING PROCEDURE
END

Figure 1 (Continued). Flow Diagram for the Adams Method.

A user interested in an efficient production program and desiring to eliminate the unnecessary moving of data resulting from the formerly separate arrays, as well as other easily discovered minor inefficiencies, should find the coordination of notation, identifiers, and labels between the preceding discussion and the following flow diagram and program listing sufficient to guide him in the task. Further, the coordination of the notation of this report with that of [5] should enable him to investigate the theory of the method of Adams with a minimum of effort expended on trivial translation.

The procedure ADAMS makes efficient use of B-5500 Algol under the restriction of generality with respect to order. However, in situations where only a few orders are needed, the running time can be decreased considerably by duplicating certain sections of the program for each order, using separate identifiers to replace much of the indexing and a switch to select the proper section of programming for a given order. Little investigation of the amount of saving achieved in this way has been done, and an evaluation of the potential gain should be profitable. This investigation might also include determination of the tradeoff between storage space and running time when a large number of orders is required.

Although the procedure ADAMS has now been tested on a wide range of equations, orders, and required accuracies with the existing step size controls, little has been done to determine the increases in efficiency to be obtained by varying the method of control and how the effects of such variation may depend upon order and required accuracy. The indication is that the existing controls produce considerably more accuracy than intended. This is particularly true when high accuracy is required at higher orders, where the penalty in running time is greatest and the largest variation in step size has been observed. Repeated step size expansions and contractions of as much as 1024 to 1 have occurred. Even a slight relaxation of the requirements for expanding step size should produce dramatic decreases in running time. Determination of a way to do this safely should prove highly worthwhile. There is slight evidence that, while the error increases with increases in the factor GR at lower orders as might be expected intuitively, this effect apparently can reverse at higher orders. A study of this phenomenon could conceivably provide information useful in improving the efficiency of the step size controls.

```
PROCEDURE ADAMS(N,XI,XF,Y,F,P,Q,DX,EA,ER,ADAMSCOEFF,RKSFNS,RKSORDER,
RKSCOEFF,START,SHANKS);
VALUE N,XI,XF,P,Q,DX,RKSFNS,RKSORDER;
REAL XI,XF,P,DX;
INTEGER N,Q,RKSFNS,RKSORDER;
REAL ARRAY Y,EA,ER,ADAMSCOEFF,RKSCOEFF[0];
PROCEDURE F,SHANKS;
INTEGER PROCEDURE START;

BEGIN
  DEFINE YI=Y#,YB=Y#,YF=Y#,POINTS=
  BEGIN
    IF NOT BGOOD THEN ALLI YB[I]←YD[I];
    X←XF-INTERVAL×(C2/C1);
    MULT←START(N,XI,XF,C1,EA,ER,F,QMINUS1,X,YB,FH,FH,YB,J,QT2M1,0,P,
    RKSFNS,RKSCOEFF);
    ALLI C[I]←YP[I]←YB[I];
    BGOOD←TRUE;

  END#,CALLOB=
  BEGIN
    SHANKS(N,X,XF,YB,F,RKSFNS,RKSORDER,RKSCOEFF,P,EA,ER,XF-X);

  END#,ALLMU=FOR MU←0STEP 1UNTIL QMINUS1 DO#,ALLI=FOR I←1STEP 1UNTIL N
  DO#,RESET=
  BEGIN
    DC←0;
    PC←Q;
    CU←(C1*(-P))×GR;
    H←INTERVAL/C1;
    ALLMU
    BEGIN
      HB[MU]←B[MU]×H;
      HBS[MU]←BS[MU]×H;

    END;
    HBSQZ←BSQZ×H;
    ALLI
    BEGIN
      EAL[I]←(EAU[I]←EA[I]×CU)×C2MQP5;
    ERL[I]←(ERU[I]←ER[I]×CU)×C2MQP5;                                00040000
    HAL[I]←ABS(EAL[I]/HBSQZ);                                       00041000
    HRL[I]←ABS(ERL[I]/HBSQZ);                                       00042000
                                                                    00043000
  END;                                                              00044000
                                                                    00045000
END#;                                                               00046000
LABEL RESTART,NEXTSTEP,FLIP,FLOP,TEST,DUBBLE,HALF,FINISH;           00047000
REAL ARRAY FH[0:(2×Q-2),0:N],B,BS,HB,HBS[0:Q-1],C,YP,YC,YD,FP,FC,EAU, 00048000
EAL,ERU,ERL,HAL,HRL[0:N];                                           00049000
REAL H,X,CU,C2,GR,YCI,BSQZ,FHJI,FMUI,HBMU,M1DP,HBSMU,HBSQZ,ERROR,   00050000
CHANGE,C2MQP5,INTERVAL;                                             00051000
INTEGER I,J,K,C1,DC,PC,MU,MULT,JZERO,QT2M1,QMINUS1,QTIMES2;         00052000
BOOLEAN BGOOD,FLIPPED,TOOSMALL;                                     00053000
C2MQP5←1/2*(Q+5);                                                   00054000
QMINUS1←Q-1;                                                        00055000
QTIMES2←Q+Q;                                                        00056000
QT2M1←QTIMES2-1;                                                    00057000
ALLMU                                                               00058000
BEGIN                                                               00059000
  B[MU]←ADAMSCOEFF[MU];                                             00060000
  BS[MU]←ADAMSCOEFF[MU+Q];                                          00061000
                                                                    00062000
END;                                                                00063000
BSQZ←ADAMSCOEFF[QTIMES2];                                           00064000
GR←ADAMSCOEFF[QTIMES2+1];                                           00065000
C1←1;                                                               00066000
H←INTERVAL←XF-XI;                                                   00067000
DX←ABS(DX);                                                         00068000
WHILE ABS(H)>DX OR C1<Q DO                                          00069000
BEGIN                                                               00070000
  C1←C1+C1;                                                         00071000
  H←INTERVAL/C1;                                                    00072000
                                                                    00073000
END;                                                                00074000
C2←C1;                                                              00075000
JZERO←J←0;                                                          00076000
F(N,XI,YI,FH[0,*]);                                                 00077000
X←XI;                                                               00078000
ALLI YB[I]←YI[I];                                                   00079000
```

```
BGOOD←TRUE;
RESTART:POINTS;
C1←C1×MULT;
C2←C2×MULT-Q;
IF(J←JZERO-1)<0THEN J←J+QT2M1;
RESET;
NEXTSTEP:X←XF-C2×H;
ALLMU
BEGIN
  IF(J←J+1)=QT2M1 THEN J←0;
  HBMU←HB[MU];
  HBSMU←HBS[MU];
  ALLI
  BEGIN
    YP[I]←(FMUI←FH[J,I])×HBMU+YP[I];
    C[I]←HBSMU×FMUI+C[I];

  END;

END;
F(N,X,YP,FP);
FLIP:ALLI YC[I]←FP[I]×HBSQZ+C[I];
F(N,X,YC,FC);
ALLI IF(CHANGE←ABS(FC[I]-FP[I]))>HAL[I]THEN IF CHANGE>ABS(HRL[I]×FC[I]
)THEN GO TO FLOP;
FLIPPED←TRUE;
GO TO TEST;
FLOP:ALLI YC[I]←FC[I]×HBSQZ+C[I];
F(N,X,YC,FP);
ALLI IF(CHANGE←ABS(FP[I]-FC[I]))>HAL[I]THEN IF CHANGE>ABS(HRL[I]×FP[I]
)THEN GO TO FLIP;
FLIPPED←FALSE;
TEST:IF(J←J+1)=QT2M1 THEN J←0;
TOOSMALL←TRUE;
ALLI
BEGIN
  FHJI←FH[J,I]←IF FLIPPED THEN FC[I]ELSE FP[I];
  ERROR←ABS(YP[I]-(YCI←C[I]←YP[I]←(FHJI×HBSQZ+C[I])));
  IF BGOOD THEN YD[I]←YCI ELSE YB[I]←YCI;
  YCI←ABS(YCI);
  IF ERROR>EAU[I]THEN IF ERROR>ERU[I]×YCI THEN GO TO HALF;          00120000
  IF ERROR>EAL[I]THEN IF ERROR>ERL[I]×YCI THEN TOOSMALL←FALSE;      00121000
                                                                    00122000
END;                                                                00123000
PC←PC+1;                                                            00124000
IF BGOOD THEN BGOOD←FALSE ELSE BGOOD←TRUE;                          00125000
IF C2≥1.0THEN C2←C2-1.0ELSE GO TO FINISH;                           00126000
IF TOOSMALL THEN                                                    00127000
BEGIN                                                               00128000
  DC←DC+1;                                                          00129000
  IF DC≥3THEN IF PC≥QT2M1 THEN IF C2≥1THEN GO TO DUBBLE;            00130000
  IF(JZERO←(J←JZERO)+1)=QT2M1 THEN JZERO←0;                         00131000
  GO TO NEXTSTEP;                                                   00132000
                                                                    00133000
END;                                                                00134000
DC←0;                                                               00135000
IF(JZERO←(J←JZERO)+1)=QT2M1 THEN JZERO←0;                           00136000
GO TO NEXTSTEP;                                                     00137000
DUBBLE:C1←C1 DIV 2;                                                 00138000
C2←(C2-1.0)/2.0;                                                    00139000
RESET;                                                              00140000
K←J;                                                                00141000
FOR MU←1STEP 1UNTIL QMINUS1 DO                                      00142000
BEGIN                                                               00143000
  IF(J←J-1)<0THEN J←J+QT2M1;                                        00144000
  IF(K←K-2)<0THEN K←K+QT2M1;                                        00145000
  ALLI FH[J,I]←FH[K,I];                                             00146000
                                                                    00147000
END;                                                                00148000
IF(J←(JZERO←J)-1)<0THEN J←J+QT2M1;                                  00149000
GO TO NEXTSTEP;                                                     00150000
HALF:IF(J←J-1)<0THEN J←J+QT2M1;                                     00151000
JZERO←J;                                                            00152000
C1←C1+C1;                                                           00153000
IF(C2←C2+C2÷2.0)<Q THEN GO TO FINISH;                               00154000
GO TO RESTART;                                                      00155000
FINISH:IF NOT FLIPPED THEN ALLI FC[I]←FP[I];                        00156000
IF NOT BGOOD THEN ALLI YB[I]←YD[I];                                 00157000
X←XF-INTERVAL×(C2/C1);                                              00158000
IF C2≠0THEN CALLOB;                                                 00159000
```

```
  ALLI YF[I]←YB[I];                                                        00160000
                                                                           00161000
END;                                                                       00162000
                                                                           00163000
```

B. The Method of Stetter, Gragg, and Butcher

1. Description of the Method

Following is a discussion of a method for the numerical integration of ordinary differential equations described by J. C. Butcher [12] in a paper titled "A Modified Multistep Method for the Numerical Integration of Ordinary Differential Equations" which appeared in the January, 1965 issue of the Journal of the Association for Computing Machinery. In this paper, Butcher presents a modification to the multistep process such that for $k \leqq 7$(where k = the number of steps) processes of order 2k + 1 are available.

A large number of possible multistep methods exist for the numerical integration of the differential equation

$$\frac{dy}{dx} = f(x,y) \ , \ y(x_o) = y_o . \tag{1-1}$$

Such methods are usually characterized by an integer k and a set of constants α_1, α_2 ---, α_k, β_o, β_1, ---, β_k. A solution is first found for the variable y at a set of points x_1, x_2, ---, x_{k-1}, (where $x_i = x_o + ih$) and thereafter by the formula:

$$\begin{aligned} y_n &= \alpha_1 y_{n-1} + \alpha_2 y_{n-2} + \text{---} + \alpha_k y_{n-k} \\ &+ h(\beta_o f_n + \beta_1 f_{n-1} + \text{---} + \beta_k f_{n-k}) \end{aligned} \tag{1-2}$$

for n = k, k + 1, --- where $y_i = y(x_i)$ and $f_i = f(x_i, y_i)$. Dahlquist [3] has shown that if the parameters α and β are chosen under a condition of stability, the order of a method cannot exceed k + 1 (if k is odd) or k + 2 (if k is even).

A modification to this process is presented by Butcher which consists of the addition to the right-hand side of equation (1-2) of an extra term

$h \beta f_{n-\theta}$ where β and θ are additional parameters to be chosen. The modified formula has the form:

$$y_n = \alpha_1 y_{n-1} + \alpha_2 y_{n-2} + --- + \alpha_k y_{n-k}$$

$$+ h (\beta f_{n-\theta} + \beta_o f_n + \beta_1 f_{n-1} + --- + \beta_k f_{n-k}) \qquad (1\text{-}3)$$

A procedure for choosing the coefficients is presented by Butcher. The simplest stable processes are for $k = 1,2,3$ with $\theta = 1/2$ and for $k = 4,5,6$ with $\theta = 1/3$. A stable process also exists for $k = 7$ with $\theta = 13/40$.

The method for implementing the formulas is to estimate $y_{n-\theta}$ and y_n using appropriate predictor formulas, then use these predicted values to evaluate the right-hand side of equation (1-3). The forms of the predictor formulas used are:

$$y_{n-\theta} = A_1 y_{n-1} + A_2 y_{n-2} + --- + A_k y_{n-k}$$

$$+ h (B_1 f_{n-1} + B_2 f_{n-2} + --- + B_k f_{n-k}) \quad (1\text{-}4)$$

$$y_n = a_1 y_{n-1} + a_2 y_{n-2} + --- + a_k y_{n-k}$$

$$+ h (b\ f_{n-\theta} + b_1 f_{n-1} + b_2 f_{n-2} + --- + b_k f_{n-k}) \quad (1\text{-}5)$$

To use this process, $y_{n-\theta}$ is first estimated using equation (1-4). The value of the function is then determined for $y_{n-\theta}$, and these two results are used in equation (1-5) to determine a value for y_n. The value of the function is then determined for y_n and a final value is then estimated using equation (1-3).

2. The Computer Procedure

A single-precision ALGOL Procedure was written to implement the integration procedure described above on the Burroughs B-5500 Computer. The procedure was designed to be used with the driver program described elsewhere in this report, but it is conceivable that it could be used with other appropriate driver programs. The procedure was written to integrate a system of differential equations each of which has the form:

$$\frac{dy}{dx} = f(x,y),\ y(x_o) = y_o.$$

Since the integration procedure described by Butcher is a multistep process, it must at all times have a history of back points. The process is, therefore, not self starting; it must rely on some other process to develop the first k steps. The starting procedure used in this implementation is a basic Runge-Kutta procedure as modified by E. B. Shanks and is discussed in paragraph E of this chapter. The starting procedure is called at the beginning of an integration and whenever it is necessary to reduce the step-size.

The step-size control is based on the difference between a predictor and a corrector; the control allows for halving and doubling of the step-size only. Equation (1-5) is used as the predictor (y_{np}) and equation (1-3) is considered to be the corrector (y_{nc}). An estimate of the magnitude of the error in a step is given by the absolute value of the difference in these two quantities. This is used in conjunction with a relative error term

ER and an absolute error term EA in the following manner: if

$$\left|\vec{y}_{np} - \vec{y}_{nc}\right| > (\vec{EA}) \left(\frac{DX}{XF - XI} \right)^{EX}$$

and

$$\left|\vec{y}_{np} - \vec{y}_{nc}\right| > \left|(\vec{ER})(\vec{y}_{nc})\right| \left(\frac{DX}{XF - XI} \right)^{EX}$$

then the step is rejected and the starting procedure is entered with the previous point and a step-size equal to half the old step-size. If

$$\left|\vec{y}_{np} - \vec{y}_{nc}\right| < (\vec{EA}) \left(\frac{DX}{XF - XI} \right)^{EX} \left(\frac{1}{2^{2k + 4}} \right)$$

or

$$\left|\vec{y}_{np} - \vec{y}_{nc}\right| < \left|(\vec{ER}) \left(\frac{DX}{XF - XI} \right)^{EX} (\vec{y}_{nc})\right| \left(\frac{1}{2^{2k + 4}} \right)$$

for three steps (without an intervening halving of the step-size) and if there is sufficient history of back points, then the step is accepted and the step-size is doubled. If neither the conditions for halving nor the conditions for doubling are met, then the step is accepted and the step-size remains constant. It is important to note that the above criteria must be satisfied for all corresponding components of the vector quantities before the conditions are considered to be met.

The method of ending the integration procedure is to run until the value of the independent variable plus the next step is either equal to or greater than the given final value i.e.

$$X + DX \geq XF.$$

If it is exactly equal, then the procedure takes one more step and quits. If X + DX > XF, then a special ending procedure is called to take the final step. This ending procedure is also a basic Runge-Kutta procedure as modified by E. B. Shanks. It is discussed in paragraph D of this chapter. The procedure call for the Butcher procedure must be as follows:

BUTCHER (N, XI, XF, K, EA, ER, DX, CON, FUNCTION, EX, RKC, START, SHANKS, YIV,
    RKSNF, RKSODR);

N - the number of dependent variables

XI - the initial value of the independent variable

XF - the final value of the independent variable

K - the number of steps to be used in the Butcher method

EA - the acceptable absolute error vector contained in an array of dimension N

ER - the acceptable relative error vector contained in an array of dimension N

DX - the suggested initial step-size

CON - the array row containing the Butcher constants required for the order of the method specified

FUNCTION - the name of the user's function evaluation procedure

EX - the error exponent

RKC - the array containing the Runge-Kutta constants

SHANKS- the name of the ending procedure

START- the name of the starting procedure

YIV - the initial values of the dependent variable; upon exiting the Butcher procedure, this array will contain the final values of the dependent variables

RKSNF- the number of function evaluations in the Runge-Kutta Shanks procedure

RKSODR - the order of the Runge-Kutta Shanks procedure

3. Flow Diagram and Program Listing

Figure 2 is the flow diagram for the method of Stetter, Gragg, and Butcher. A listing of the program is given at the end of this section.

4. Results and Conclusions

The following remarks will be directed to the problem of step-size control in the Butcher procedure. Although the method of step-size control, as described previously, was adequate for the purposes of this project, it is felt that some improvement is desirable. The difficulty observed was that in virtually all cases the accuracy achieved by the procedure was one to two orders of magnitude greater than the accuracy asked. For the third order Butcher process ($k = 1$), the step-size control method is completely unsatisfactory yielding long running times and accuracies as much as four times greater than those asked. The process in this case is essentially a third-order Runge-Kutta process. The equation used as a predictor in the step-size control scheme simply is not accurate enough in this case; this results in relatively large differences between the predictor and the corrector.

In order to improve the relation between the accuracy asked of the procedure and the accuracy achieved, it is desirable to study ways of improving the step-size control of the Butcher procedure. Such an improvement should also result in a faster running time for the method. One possible way of improving the step-size control is to use some form of two-step/one-step comparison. This could be accomplished in the Butcher process by using a predictor and corrector of the same order where the corrector uses alternate points of the history and a step-size twice as large as that of the predictor. The use of twice the step-size has the advantage of not requiring the recomputation of back points.

The investigation of new methods of step-size control should be done in double precision so that all of the orders of the Butcher process can be investigated. If single precision were used, only lowest orders of the process could be adequately investigated. This would not give a complete picture of the operation of the Butcher process.

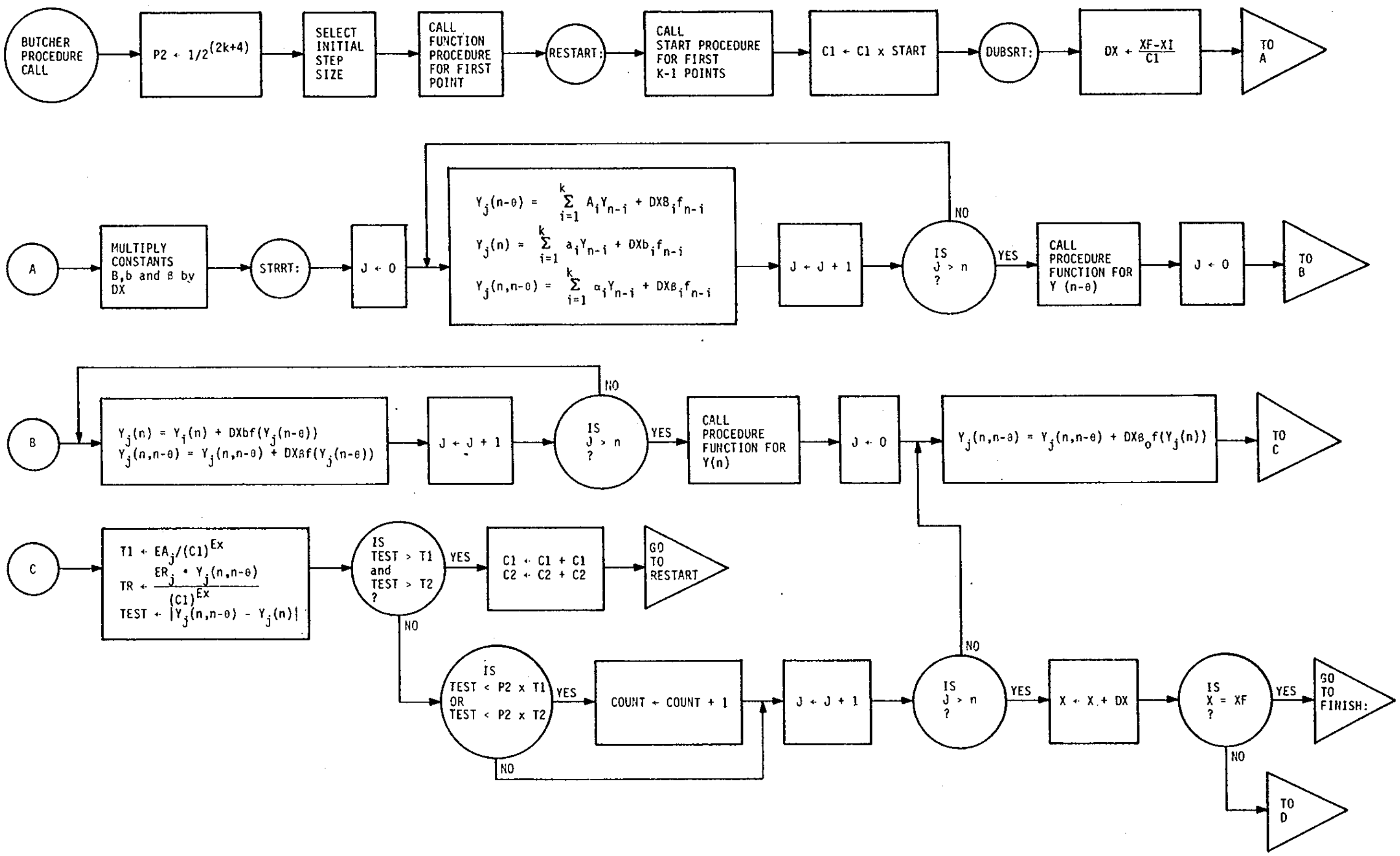
BUTCHER PROCEDURE CALL
P2 ← 1/2^(2k+4)
SELECT INITIAL STEP SIZE
CALL FUNCTION PROCEDURE FOR FIRST POINT
RESTART:
CALL START PROCEDURE FOR FIRST K-1 POINTS
C1 ← C1 x START
DUBSRT:
DX ← (XF-XI)/C1
TO A
A
MULTIPLY CONSTANTS B,b and β by DX
STRRT:
J ← 0
Yj(n-θ) = Σ(i=1..k) AiYn-i + DXBifn-i
Yj(n) = Σ(i=1..k) aiYn-i + DXbifn-i
Yj(n,n-θ) = Σ(i=1..k) αiYn-i + DXβifn-i
J ← J + 1
IS J > n ?
NO
YES
CALL PROCEDURE FUNCTION FOR Y (n-θ)
J ← 0
TO B
B
Yj(n) = Yi(n) + DXbf(Yj(n-θ))
Yj(n,n-θ) = Yj(n,n-θ) + DXβf(Yj(n-θ))
J ← J + 1
IS J > n ?
NO
YES
CALL PROCEDURE FUNCTION FOR Y(n)
J ← 0
Yj(n,n-θ) = Yj(n,n-θ) + DXβ0f(Yj(n))
TO C
C
T1 ← EAj/(C1)^Ex
TR ← ERj • Yj(n,n-θ) / (C1)^Ex
TEST ← |Yj(n,n-θ) - Yj(n)|
IS TEST > T1 and TEST > T2 ?
YES
NO
C1 ← C1 + C1
C2 ← C2 + C2
GO TO RESTART
IS TEST < P2 x T1 OR TEST < P2 x T2
YES
NO
COUNT ← COUNT + 1
J ← J + 1
IS J > n ?
NO
YES
X ← X + DX
IS X = XF ?
YES
NO
GO TO FINISH:
TO D

Figure 2. Flow Diagram for the Stetter-Gragg-Butcher Method.

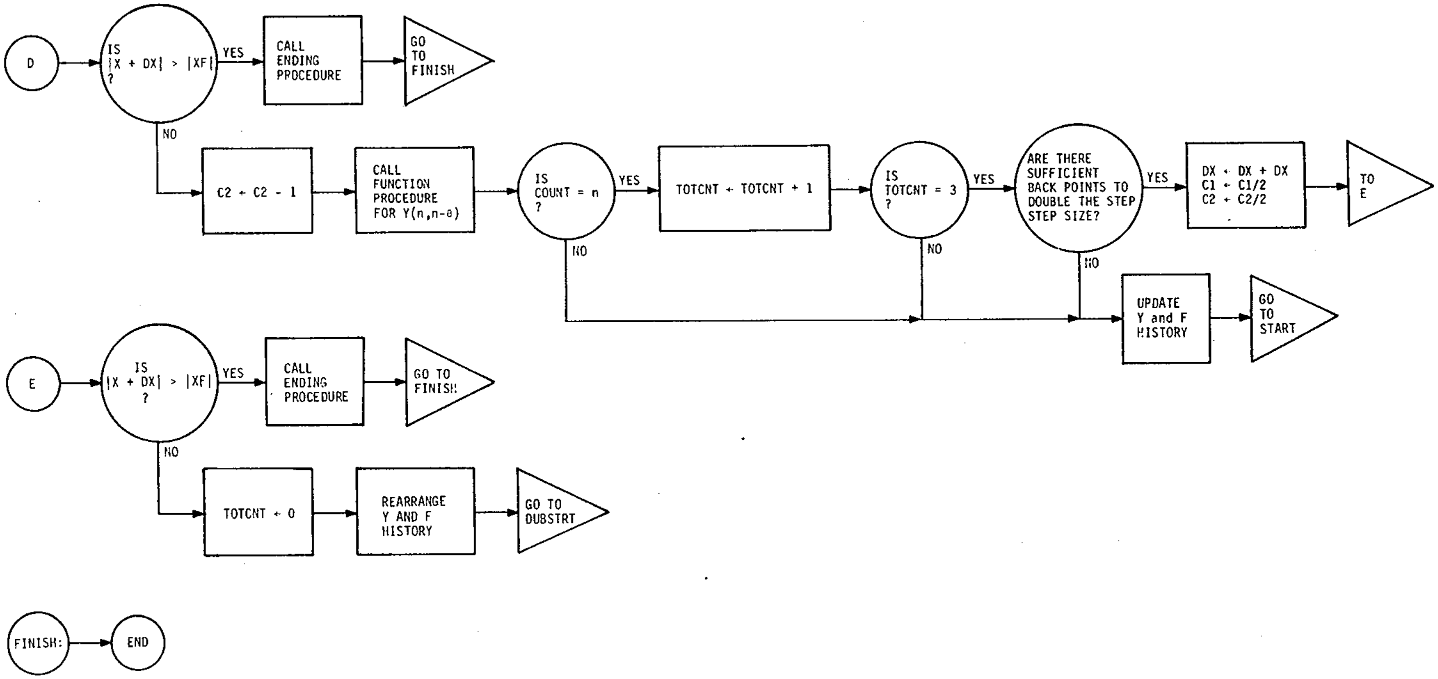

Figure 2 (Continued). Flow Diagram for the Stetter-Gragg-Butcher Method.

```
PROCEDURE BUTCHER(N,XI,XF,K,EA,ER,DX,CON,FUNCTION,EX,RKC,START,SHANKS,
YIV,RKSNF,RKSODR);
VALUE N,XI,XF,K,DX,CON,EX,RKSNF,RKSODR;
REAL ARRAY YIV[0];
INTEGER RKSNF,RKSODR;
INTEGER N,K;
PROCEDURE FUNCTION,SHANKS;
INTEGER PROCEDURE START;
REAL XI,XF,DX,EX;
REAL ARRAY RKC[0];
REAL ARRAY CON,EA,ER[0];

BEGIN
  REAL ARRAY Y,F[0:16,0:N];
  REAL SC1,X;
  REAL DX2;
  REAL DX1,COA,COB,COLA,COLB,COGA,COGB,TEST,TEMPY,TEMPF,A1,A2,A3,C2;
  INTEGER I,J,CYL,INDEX,C1,M;
  INTEGER CYL3;
  REAL ARRAY SUMYIP,SUMYP,SUMYC,FV1[0:N];
  LABEL STRRT,RESTART,FINISH;
  REAL P2,T1,T2;
  INTEGER COUNT,TOTCNT,CYL1,CYL2,M1;
  LABEL DUBSRT;
  REAL ARRAY COO[0:3×K];
  INTEGER CYO;
  INTEGER COUNTER;
  INTEGER KM1;
  REAL OMT,K6,K61,K62;
  REAL INTV;
  INTEGER KM3,J2,J3,J6;
  REAL XDXT,XDX;
  REAL ARRAY RE,AE[0:N];
  FOR I←1STEP 1UNTIL N DO Y[0,I]←YIV[I];
  IF K=1OR K=2OR K=3THEN OMT←0.5ELSE OMT←2/3;
  K6←6×K;
  K61←(6×K)+1;
  K62←(6×K)+2;
  KM1←K-1;
  INTV←XF-XI;
```

```
X←XI;
C1←1;
WHILE(C1<K+1)OR((ABS(INTV)/C1)>ABS(DX))DO C1←C1+C1;
C2←C1;
P2←1/(2*((2×K)+4));
CYL←0; ·
CYO←0;
TOTCNT←0;
FUNCTION(N,XI,Y[0,*],F[0,*]);
RESTART:COUNTER←KM1;
C1←C1×(I←START(N,XI,XF,C1,EA,ER,FUNCTION,KM1,X,YIV,Y,F,YIV,CYO,16,2,EX
,RKSNF,RKC));
C2←C2×I-KM1;
CYL←CYO;
DUBSRT:DX←INTV/C1;
KM3←3×KM1;
FOR J←0STEP 3UNTIL KM3 DO
BEGIN
  J2←2×J;
  COO[J]←CON[J2+1]×DX;
  COO[J+1]←CON[J2+3]×DX;
  COO[J+2]←CON[J2+5]×DX;

END;
SC1←C1*EX;
FOR I←1STEP 1UNTIL N DO
BEGIN
  AE[I]←EA[I]/SC1;
  RE[I]←ER[I]/SC1;

END;
A1←CON[K6]×DX;
A2←CON[K61]×DX;
A3←CON[K62]×DX;
STRRT:XDXT←X+(DX×OMT);
XDX←X+DX;
FOR I←1STEP 1UNTIL N DO SUMYIP[I]←SUMYP[I]←SUMYC[I]←0;
FOR J←0STEP 1UNTIL KM1 DO
BEGIN
  CYL3←(KM1-J+CYL)MOD 16;
```

```
    J3←3×J;
    J6←6×J;
    COA←CON[J6];
    COB←COO[J3];
    COLA←CON[J6+2];
    COLB←COO[J3+1];
    COGA←CON[J6+4];
    COGB←COO[J3+2];
    FOR I←1STEP 1UNTIL N DO
    BEGIN
      TEMPY←Y[CYL3,I];
      TEMPF←F[CYL3,I];
      SUMYIP[I]←SUMYIP[I]+(COA×TEMPY)+(COB×TEMPF);
      SUMYP[I]←SUMYP[I]+(COLA×TEMPY)+(COLB×TEMPF);
      SUMYC[I]←SUMYC[I]+(COGA×TEMPY)+(COGB×TEMPF);

    END;

  END;
  FUNCTION(N,XDXT,SUMYIP,FV1);
  FOR I←1STEP 1UNTIL N DO
  BEGIN
    TEMPF←FV1[I];
    SUMYP[I]←SUMYP[I]+(A1×TEMPF);
    SUMYC[I]←SUMYC[I]+(A2×TEMPF);

  END;
  FUNCTION(N,XDX,SUMYP,FV1);
  CYL←(CYL+1)MOD 16;
  CYO←(CYL+KM1)MOD 16;
  COUNT←0;
  FOR I←1STEP 1UNTIL N DO
  BEGIN
    TEMPY←SUMYC[I]+(A3×FV1[I]);
    T1←AE[I];
    T2←ABS(RE[I]×TEMPY);
    TEST←ABS(TEMPY−SUMYP[I]);
    IF TEST>T1 AND TEST>T2 THEN
    BEGIN
      C2←C2+C2;
    CYL←(CYL+15)MOD 16;
    CYO←(CYL+KM1)MOD 16;
    IF C2<KM1 THEN
    BEGIN
      SHANKS(N,X,XF,Y[CYO,*],FUNCTION,RKSNF,RKSODR,RKC,EX,EA,ER,DX);
      GO TO FINISH;

    END;
    C1←C1+C1;
    GO TO RESTART;

  END;
  Y[CYO,I]←TEMPY;
  IF TEST<P2×T1 OR TEST<P2×T2 THEN COUNT←COUNT+1;

END;
C2←C2-1;
X←XF-(DX×C2);
IF C2=0THEN GO TO FINISH;
IF C2<1THEN
BEGIN
  SHANKS(N,X,XF,Y[CYO,*],FUNCTION,RKSNF,RKSODR,RKC,EX,EA,ER,DX);
  GO TO FINISH;

END;
FUNCTION(N,X,Y[CYO,*],F[CYO,*]);
IF COUNT=N THEN
BEGIN
  TOTCNT←TOTCNT+1;
  IF TOTCNT≥3THEN
  BEGIN
    IF COUNTER≥2×K THEN
    BEGIN
      COUNTER←0;
      C2←C2/2;
      C1←C1/2;
      IF C2<1THEN
      BEGIN
        SHANKS(N,X,XF,Y[CYO,*],FUNCTION,RKSNF,RKSODR,RKC,EX,EA,ER,DX);
        GO TO FINISH;

        END;
        TOTCNT←0;
        FOR I←1STEP 1UNTIL N DO FOR J←1STEP 1UNTIL KM1 DO
        BEGIN
          CYL1←(CYO+16-J)MOD 16;
          CYL2←(CYO+16-(2×J))MOD 16;
          Y[CYL1,I]←Y[CYL2,I];
          F[CYL1,I]←F[CYL2,I];

        END;
        GO TO DUBSRT;

      END;

    END;

  END;
  COUNTER←COUNTER+1;
  GO TO STRRT;
  FINISH:FOR I←1STEP 1 UNTIL N DO YIV[I]←Y[CYO,I];
END BUTCHER;
```

C. The Cowell Method

1. Description of the Method

Cowell's method as described herein is a multistep predictor-corrector method for the numerical solution of the first-order vector differential equation

$$\vec{y}\,'(x) = \frac{d}{dx}\,\vec{y}(x) = \vec{f}\,(x,\vec{y}(x)), \quad \vec{y}\,(x_0) = \vec{y}_0. \tag{1-1}$$

A complete derivation and description of Cowell's method can be found in [9] and [18]; only the essential formulas are included here.

The following notation is adopted. Let q be an even positive integer, m = q/2, h be the step size (assumed to be constant over some set of calculations),

$$x_n = x_0 + nh, \; \vec{y}_n = \vec{y}\,(x_n), \text{ and } \vec{f}_n = \vec{f}\,(x_n, \vec{y}_n) \; .$$

The predictor formula is

$$\vec{y}_n = h\,[\overrightarrow{\delta^{-1}\, f_{n-\frac{1}{2}}} + \sum_{j=0}^{q} P_j\, \vec{f}_{n-1-j}]\,, \tag{1-2}$$

The corrector formula is

$$\vec{y}_n = h\,[\overrightarrow{\delta^{-1}\, f_{n-\frac{1}{2}}} + \sum_{j=0}^{q} C_j\, \vec{f}_{n-j}]\,, \tag{1-3}$$

and the mid-range formula is

$$\vec{y}_n = h\,[\overrightarrow{\delta^{-1}\, f_{n-\frac{1}{2}}} + \sum_{j=0}^{q} M_j\, \vec{f}_{n+m-j}]\,. \tag{1-4}$$

The predictor formula gives $\vec{y}_n$ in terms of $\overrightarrow{\delta^{-1} f_{n-\frac{1}{2}}}$ and the function values at the previous q+1 points; the corrector formula gives a new value of $\vec{y}_n$ in terms of $\overrightarrow{\delta^{-1}f}_{n-\frac{1}{2}}$, the old value of $\vec{y}_n$, and the function values at the previous q points; the mid-range formula gives a value of $\vec{y}_n$ in terms of $\overrightarrow{\delta^{-1}f}_{n-\frac{1}{2}}$ and the function values at the q+1 consecutive points centered around x_n.

The equation

$$\overrightarrow{\delta^{-1}f}_{n-\frac{1}{2}} = \overrightarrow{\delta^{-1}f}_{n-1-\frac{1}{2}} + \vec{f}_{n-1} \qquad (1\text{-}5)$$

completes the set of formulas necessary for the numerical solution of (1-1).

If it is assumed that

$$\left\{\vec{f}_i\right\}_{i=0}^{q} \quad \text{and} \quad \vec{y}_m$$

have been obtained by some starting procedure, the mid-range formula (1-4) can be applied with n=m to obtain

$$\overrightarrow{\delta^{-1}f}_{m-\frac{1}{2}} .$$

Equation (1-5) can then be applied m times to obtain

$$\overrightarrow{\delta^{-1}f}_{q-\frac{1}{2}} .$$

For each positive integer i

$$\overrightarrow{\delta^{-1}f}_{q+i-\frac{1}{2}}$$

can be computed from

$$\overrightarrow{\delta^{-1}f}_{q+i-1-\frac{1}{2}}$$

and $\vec{f}_{q+i-1}$ using (1-5); $\vec{y}_{q+i}$ can be computed using the predictor (1-2); $\vec{f}_{q+i}$ can be computed from the predicted value; $\vec{y}_{q+i}$ can be computed using the corrector (1-3); $\vec{f}_{q+i}$ can be computed from the corrected value; if necessary, iteration can be resorted to, using (1-3), until the last two computed values of $\vec{y}_{q+i}$ agree to sufficient accuracy. For any $j \geq m$ a value of $\vec{y}_{q+j-m}$ can be obtained from the mid-range formula (1-4) and compared with the value obtained from the predictor-corrector step. If the two values of $\vec{y}_{q+j-m}$ are in sufficient agreement, the values up through $\vec{y}_{q+j}$ are considered acceptable; if not, $\vec{y}_{q+j-m}$ is considered the last acceptable value and all values beyond are rejected.

Hence, the knowledge of (1-2), (1-3), (1-4), and (1-5) is sufficient to apply Cowell's method in the numerical solution of (1-1). The coefficients $\{P_j\}_{j=0}^{q}$, $\{C_j\}_{j=0}^{q}$, and $\{M_j\}_{j=0}^{q}$ are given in [18] for $q=4$, 6, 8, 10, 12, 14, and 16.

2. <u>The Computer Program</u>

The Cowell computer program is a Burroughs B-5500 ALGOL single-precision procedure whose declaration is as follows:

```
procedure Cowell (m, xi, xf, y, f, ea, er, p, dx, rksfn,
                  rksorder, rkscoeff, q, cowellcoeff, start, shanks);
value n, xi, xf, p, dx, rksfn, rksorder, q;
integer n, rksfn, rksorder, q;
real xi, xf, p, dx;
real array y, ea, er, rkscoeff, cowellcoeff [0];
procedure f, shanks;
integer procedure start;
```

The parameters of the procedure are defined as follows:

n - the number of dependent variables in the vectors $\vec{y}$ and $\vec{f}$

xi - x_o, the starting value of the independent variable x

xf - the final value of the independent variable x

y - the array in which $\vec{y}_o = \vec{y}$ (xi) is located upon entry and in which $\vec{y}$ (xf) is located upon exit

f - the procedure which computes $\vec{f} = \vec{f}(x, \vec{y})$

ea - the array containing the absolute error vector

er - the array containing the relative error vector

p - the exponent used in step size control

rksfn - the number of function evaluations used in the Runge-Kutta-Shanks starting and closing procedures

rksorder - the order of the Runge-Kutta-Shanks closing procedure

rkscoeff - the array containing the Runge-Kutta-Shanks coefficients for the starting and closing procedures.

q - the even integer used in describing Cowell's method

cowellcoeff - the array containing the Cowell coefficients

start - the starting procedure

shanks - the closing procedure.

The procedure performs the numerical integration of (1-1) from x = xi to x = xf. The step size h used is always the length of the interval xf - xi divided by a power of 2 in order to avoid error building in the independent variable two counters, c1 and c2 are kept. c1 is always a positive, integral power of 2, and h = (xf - xi)/c1. c2 is the number of steps necessary to step from the present x to xf using the current step size h. Initially c2 = c1; as each step is taken c2 is decremented by one and the present value of x is computed by x = xf - h c2. If h is halved, c1 and c2 are doubled; if h is doubled, c1 and c2 are halved. Hence c2 need not be integral.

The error vectors $\vec{ea}$ and $\vec{er}$, like $\vec{y}$, have n components. (Although the base of the arrays y, ea, and er is zero, the n components are placed in positions 1, 2, ..., n of the arrays.) The procedure's error control attempts to guarantee that, in integrating from xi to xf, each component of $\vec{y}$ will not be in absolute error more than the corresponding component of $\vec{ea}$ and will not be in relative error more than the corresponding component of $\vec{er}$. At each step, the procedure requires that for each i, $1 \le i \le n$, either the absolute error in y [i] does not exceed ea [i]/($c1^p$) or the relative error in y [i] does not exceed er [i]/($c1^p$).

If p = 1 and $\vec{er} = 0$ then the accumulated error in any component of $\vec{y}$ cannot exceed the corresponding component of $\vec{ea}$. If the error is assumed to accumulate randomly as the square root of the number of steps, $p = \frac{1}{2}$ and $\vec{er} = 0$ will cause the accumulated error in any component of $\vec{y}$ to be approximately the corresponding component of $\vec{ea}$.

If p = 1 and $\vec{ea} = 0$ then the accumulated error in any component of $\vec{y}$ cannot exceed the corresponding component of $\vec{er}$ times the largest value assumed by that component of $\vec{y}$ during the integration. If the error is assumed to accumulate randomly as the square root of the number of steps, $p = \frac{1}{2}$ and $\vec{ea} = 0$ will cause the accumulated error in any component of $\vec{y}$ to be approximately the corresponding component of $\vec{er}$ times some average value assumed by that component of $\vec{y}$ during the integration.

The procedure f which computes $\vec{f} = \vec{f}\,(x,y)$ has the following declaration:

```
procedure f(n, x, yv, fv);

value n;

integer n;

real x;

real array yv, fv [0];
```

The parameters of the procedure f are defined as follows:

n - the number of dependent variables in the vectors $\vec{y}$ and $\vec{f}$

x - the value of the independent variable

yv - the array in which $\vec{y}$ is stored

fv - the array in which $\vec{f}$ is stored after computation

The procedure start is the general multistep method starting procedure described in paragraph E of this chapter. The procedure shanks is the Runge-Kutta-Shanks integration procedure described in paragraph D of this chapter. The coefficient array rkscoeff contains the Runge-Kutta-Shanks coefficients in the order required by the procedures start and shanks. The number of function evaluations rksfn is required by both start and shanks; the order rksorder is required by shanks.

The array cowellcoeff contains the coefficients of (1-2), (1-3), and (1-4) in the order P_0, P_1, . . ., P_q, C_0, C_i, . . ., C_q, M_0, M_1, . . ., M_q; P_0 is in the zero position of the array.

The suggested initial step size dx is optional. The procedure first sets cl = 2 and doubles cl until cl ≥ q. If dx = 0 or dx ≠ 0 and $h \leq |dx|$ then cl is left alone. Otherwise, cl is doubled until $h \leq |dx|$.

The integration now begins.

$$\vec{f}_0 = \vec{f}(x_0, \vec{y}_0)$$

is computed. The start procedure is called to obtain

$$\left\{f_i\right\}_{i=1}^{q}\ ,\quad \vec{y}_m,\ \vec{y}_q \text{ and } x_q .$$

c1 and c2 are adjusted if h was changed by the start procedure. c2 is decremented by q since q steps took place in the start procedure. If c2 < m, closing takes place. Otherwise,

$$\overrightarrow{\delta^{-1} f}_{m-\frac{1}{2}}$$

is calculated from

$$\left\{f_i\right\}_{i=0}^{q}$$

and $\vec{y}_m$ using the mid-range formula (1-4). m applications of (1-5) yield

$$\overrightarrow{\delta^{-1} f}_{q-\frac{1}{2}}$$

and n is set equal to q.

For $1 \leq i \leq m$ the following set of steps takes place. c2 is decremented by 1, and x_{n+i} is calculated.

$$\overrightarrow{\delta^{-1} f}_{n+i-\frac{1}{2}}$$

is calculated from

$$\overrightarrow{\delta^{-1} f}_{n+i-1-\frac{1}{2}}$$

and $\vec{f}_{n+i-1}$ using (1-5). $\vec{y}_{n+i}$ is calculated using the predictor (1-2), and

$\vec{f}_{n+i}$ is calculated. $\vec{y}_{n+i}$ is next calculated using the corrector (1-3), and $\vec{f}_{n+i}$ is again calculated. Let $\vec{v}$ be the vector which is the absolute value of the difference between the last two calculated values of $\vec{y}_{n+i}$. Each component of $\vec{v}$ is compared with the corresponding component of $\vec{ea}/(10 \cdot cl^p)$ for absolute error and with the product of the corresponding components of $\vec{er}/ (10 \cdot cl^p)$ and the last calculated value of $\vec{y}_{n+i}$ for relative error. If any component of $\vec{v}$ exceeds in both the absolute and the relative error tests, the steps which calculate $\vec{y}_{n+i}$ using the corrector (1-3), calculate $\vec{f}_{n+i}$ from the value of $\vec{y}_{n+i}$, and which test the last two calculated values of $\vec{y}_{n+i}$ are repeated. When each component of $\vec{v}$ does not exceed in either the absolute or the relative error test, the last values of $\vec{y}_{n+i}$ and $\vec{f}_{n+i}$ are retained.

The mid-range formula (1-4) is now used to calculate a new value of $\vec{y}_n$ from

$$\left\{f_{n+i}\right\}_{i=-m}^{m}$$

and

$$\overrightarrow{\delta^{-1} f_{n-\frac{1}{2}}}.$$

Let $\vec{v}$ be the vector which is the absolute value of the differences between the new value of $\vec{y}_n$ and the previously calculated value of $\vec{y}_n$. If sufficient history is available for doubling the step size, i.e., $n > q + m$, each component of $\vec{v}$ is compared with the corresponding component of $\vec{ea}/(10 \cdot cl^p \cdot 2^{q+3})$ for absolute error and with the product of the corresponding components of $\vec{er}/(10 \cdot cl^p \cdot 2^{q+3})$ and the new value of $\vec{y}_n$ for relative error.

If each component of $\vec{v}$ does not exceed in either the absolute or the relative error tests, the last m steps are accepted, c1 and c2 are halved, and the step size is doubled. If c2 < m, closing takes place. Otherwise

$$\left\{f_i\right\}_{i=0}^{q}$$

becomes

$$\left\{\vec{f}_{n-m+2i}\right\}_{i=0}^{q},$$

$\vec{y}_m$ becomes $\vec{y}_{n-m}$, $\vec{y}_q$ becomes $\vec{y}_{n+m}$, x_q becomes x_{n+m}, and, as if the starting procedure had calculated these values, control returns to the step where

$$\overrightarrow{\delta^{-1}f_{m-\frac{1}{2}}}$$

is calculated using the mid-range formula (1-4).

If any component of $\vec{v}$ exceeds in both the absolute and the relative error tests, this component and each untested component is compared with the corresponding component of $\vec{ea}/(10 \cdot c1^p)$ for absolute error and with the product of the corresponding components of $\vec{er}/(10 \cdot c1^p)$ and the new value of $\vec{y}_n$ for relative error. If each component of $\vec{v}$ does not exceed in either the absolute or the relative error test, the last m steps are accepted and the step size remains unchanged. If c2 < m, closing takes place. Otherwise, n becomes n + m and control returns to the steps which calculate

$$\left\{y_{n+i}\right\}_{i=1}^{m}.$$

If any component of $\vec{v}$ exceeds in both the absolute and the relative error tests, the last m steps are rejected, c2 is incremented by m, cl and c2 are doubled, and the step size is halved. $\vec{f}_0$ becomes $\vec{f}_n$, $\vec{y}_0$ becomes $\vec{y}_n$, x_0 becomes x_n, and control is returned to the step which calls the start procedure.

If sufficient history is not available for doubling, control transfers as if the first component of $\vec{v}$ exceeded both the first component of $\vec{ea}/(10 \cdot cl^p \cdot 2^{q+3})$ and the product of the first components of $\vec{er}/(10 \cdot cl^p \cdot 2^{q+3})$ with the first component of the new value of $\vec{y}_n$.

Closing takes place whenever m steps at the present step size would carry the integration beyond xf, i.e., whenever c2 < m. If c2 > 0, the Runge-Kutta-Shanks procedure is used to integrate from the present value of x to xf; if c2 = 0, the present value of x is xf. In either case, the integration is now complete.

Several efficiency measures are employed in the program. First, the coefficients

$$\left\{ P_j \right\}_{j=0}^{q} ,$$

$$\left\{ C_j \right\}_{j=0}^{q} ,$$

and

$$\left\{ M_j \right\}_{j=0}^{q}$$

are multiplied by the step size h and stored as multiplied until the step size changes. Second, the vectors $\vec{ea}/(10 \cdot cl^p)$, $\vec{er}/(10 \cdot cl^p)$, $\vec{ea}/(10 \cdot cl^p \cdot 2^{q+3})$, and $\vec{er}/(10 \cdot cl^p \cdot 2^{q+3})$ are calculated from $\vec{ea}$ and $\vec{er}$ and stored as calculated until the step size changes. Third, the corrector partial sum

$$h\overrightarrow{\delta^{-1}f}_{n-\frac{1}{2}} + h\sum_{j=1}^{q} C_j f_j$$

is computed and stored at each step; successive applications of the corrector only require adding $h \cdot C_0 \cdot f_n$ to obtain $\vec{y}_n$. Fourth, during applications of the corrector, two arrays are used to store the last two calculated values of $\vec{y}_n$; a flag is used to mark the last calculated value so that the next value is placed in the unflagged array and the flag is switched. This avoids transfer from array to array as successive corrector iterates are computed. Fifth, cyclic indexing is used to avoid moving the function value history after each step or set of steps unless doubling takes place.

One unusual condition can result. If, during any step taken in computing

$$\left\{\vec{y}_{n+i}\right\}_{i=1}^{m},$$

the number of times through the corrector exceeds eight, control transfers as if the set of m steps has been completed and rejected, i.e., a step size halving was called for with a restart beginning at $\vec{y}_n$.

3. Flow Diagram and Program Listing

Figure 3 is the flow diagram for the Cowell method. The program listing follows at the end of this section.

4. Results and Conclusions

The first important conclusion concerns the error control. The specified tolerances for absolute and relative error are handled vectorially to allow for systems in which the units of the various dependent variables are not the same. Such systems arise in physics, for example, from reduction of second order equations of motion in two dimensions to a first order system in which two variables are positions and two variables are velocities. More important, however, is the requirement at each step that the error in each variable not exceed the specified tolerances divided by cl^p, where $0 \leq p \leq 1$. If $p = 0$, conventional vectorial error control results. If $p \neq 0$, however, an interesting phenomenon occurs. As the step size decreases, higher accuracy is required; as the step size increases, less accuracy is required. Hence, halving is often required sooner after a previous halving than when $p = 0$, and halving immediately after doubling is less frequent since the increase in error due to doubling is accompanied by a decrease in accuracy required.

One major result of this error control is the linearity of error obtained as a function of error asked. Earlier experiments [18] with $p = 0$ showed that dividing the asked error by ten sometimes had little or no effect on the error obtained; dividing the error asked by two sometimes decreased the error obtained by a factor of ten. Present experiments with $p = \frac{1}{2}$ show that multiplication of the error asked by a constant usually causes the error obtained to be multiplied by the same constant.

Division of the asked error tolerances by $10 \cdot cl^p$ rather than cl^p was determined experimentally to be necessary in order to assure that the error at each step be held to its desired value. This seems to be a peculiarity of the mid-range formula type of error estimation; namely, that the actual error after each set of m steps can be as much as ten times as large as the estimate given by the mid-range test.

Doubling occurs when the estimated error is less than the asked error tolerances divided by $10 \cdot cl^p \cdot 2^{q+3}$; hence, the doubling criteria are the accepting criteria divided by 2^{q+3}. This factor was also chosen experimentally,

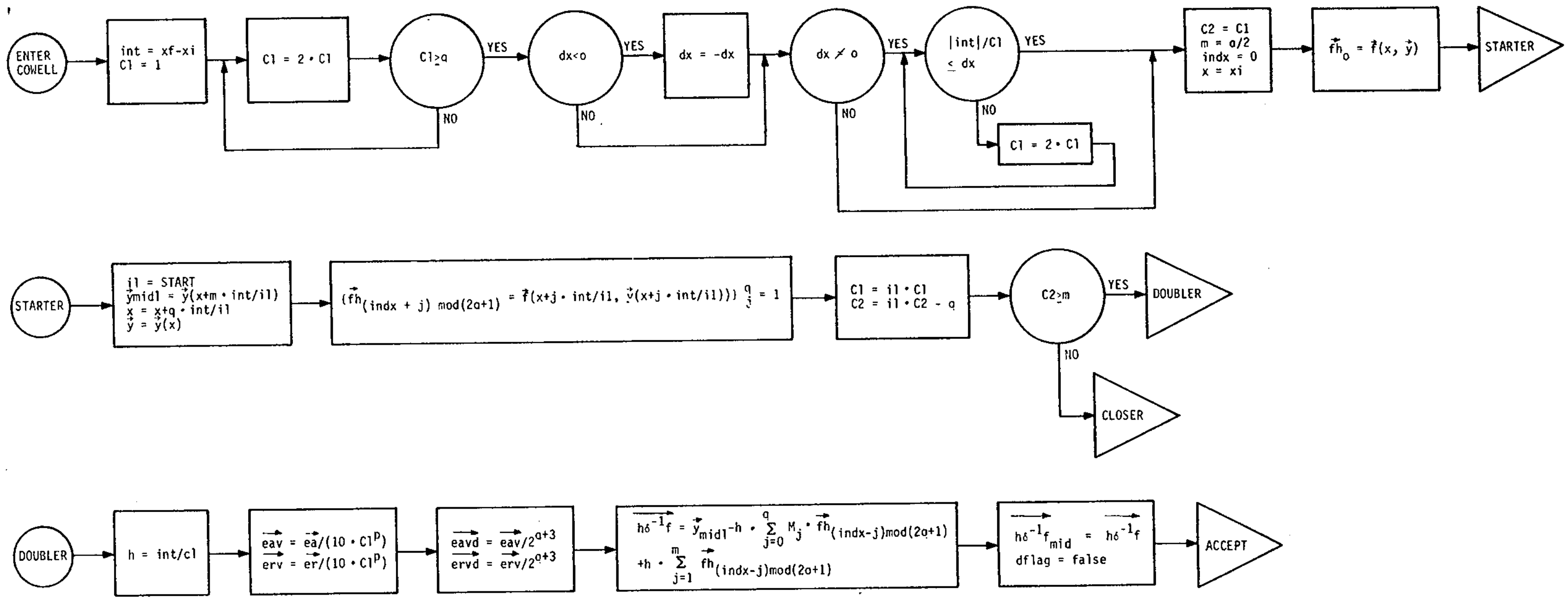
ENTER COWELL
int = xf-xi
C1 = 1
C1 = 2 • C1
C1≥α
YES
NO
dx<o
YES
NO
dx = -dx
dx ≯ o
YES
NO
|int|/C1 ≤ dx
YES
NO
C1 = 2 • C1
C2 = C1
m = α/2
indx = 0
x = xi
$\vec{fh}_o = \vec{f}(x, \vec{y})$
STARTER
STARTER
i1 = START
$\vec{y}$mid1 = $\vec{y}$(x+m • int/i1)
$\vec{x}$ = $\vec{x}$+q • int/i1
$\vec{y} = \vec{y}(x)$
$(\vec{fh}_{(indx + j) \bmod (2\alpha+1)} = \vec{f}(x+j \cdot int/i1, \vec{v}(x+j \cdot int/i1)))_{j=1}^{q}$
C1 = i1 • C1
C2 = i1 • C2 - q
C2≥m
YES
NO
DOUBLER
CLOSER
DOUBLER
h = int/c1
$\vec{eav} = \vec{ea}/(10 \cdot C1^p)$
$\vec{erv} = \vec{er}/(10 \cdot C1^p)$
$\vec{eavd} = \vec{eav}/2^{\alpha+3}$
$\vec{ervd} = \vec{erv}/2^{\alpha+3}$
$\overrightarrow{h\delta^{-1}f} = \vec{y}_{mid1} - h \cdot \sum_{j=0}^{q} M_j \cdot \vec{fh}_{(indx-j)\bmod(2\alpha+1)}$
$+h \cdot \sum_{j=1}^{m} \vec{fh}_{(indx-j)\bmod(2\alpha+1)}$
$\overrightarrow{h\delta^{-1}f_{mid}} = \overrightarrow{h\delta^{-1}f}$
dflag = false
ACCEPT

Figure 3. Flow Diagram for the Cowell Method.

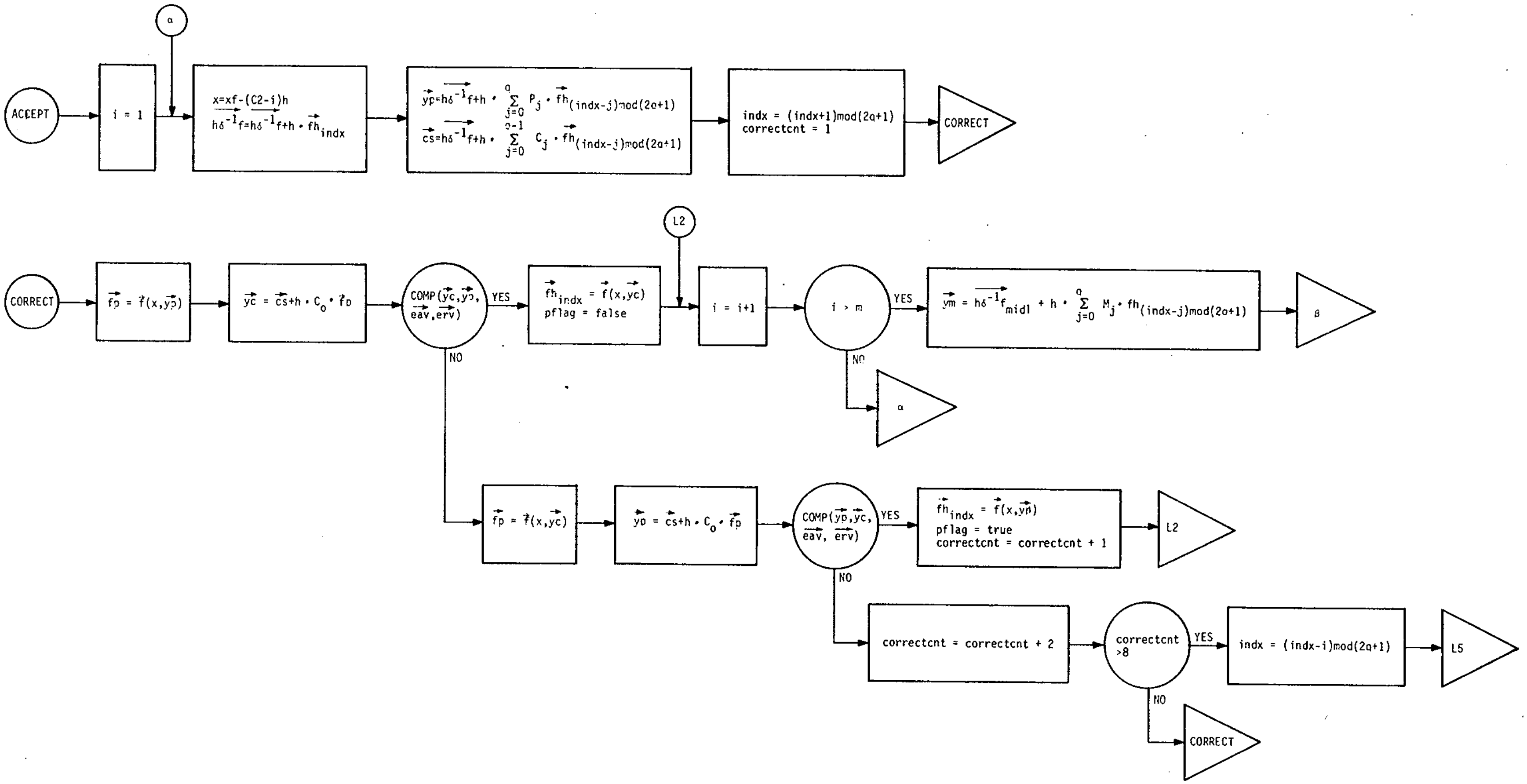
ACCEPT
i = 1
α
x=xf-(C2-i)h
$\overrightarrow{h\delta^{-1}f}=\overrightarrow{h\delta^{-1}f}+h\cdot\vec{fh}_{indx}$
$\vec{yp}=\overrightarrow{h\delta^{-1}f}+h\cdot\sum_{j=0}^{q}P_j\cdot\vec{fh}_{(indx-j)mod(2q+1)}$
$\vec{cs}=\overrightarrow{h\delta^{-1}f}+h\cdot\sum_{j=0}^{q-1}C_j\cdot\vec{fh}_{(indx-j)mod(2q+1)}$
indx = (indx+1)mod(2q+1)
correctcnt = 1
CORRECT
CORRECT
$\vec{fp}=\vec{f}(x,\vec{yp})$
$\vec{yc}=\vec{cs}+h\cdot C_o\cdot\vec{fp}$
COMP($\vec{yc},\vec{yp}$, $\vec{eav},\vec{erv}$)
YES
NO
$\vec{fh}_{indx}=\vec{f}(x,\vec{yc})$
pflag = false
L2
i = i+1
i > m
YES
NO
$\vec{ym}=\overrightarrow{h\delta^{-1}f_{midl}}+h\cdot\sum_{j=0}^{q}M_j\cdot fh_{(indx-j)mod(2q+1)}$
β
α
$\vec{fp}=\vec{f}(x,\vec{yc})$
$\vec{yp}=\vec{cs}+h\cdot C_o\cdot\vec{fp}$
COMP($\vec{yp},\vec{yc}$, $\vec{eav},\vec{erv}$)
YES
NO
$\vec{fh}_{indx}=\vec{f}(x,\vec{yp})$
pflag = true
correctcnt = correctcnt + 1
L2
correctcnt = correctcnt + 2
correctcnt >8
YES
NO
indx = (indx-i)mod(2q+1)
L5
CORRECT

Figure 3 (Continued). Flow Diagram for the Cowell Method.

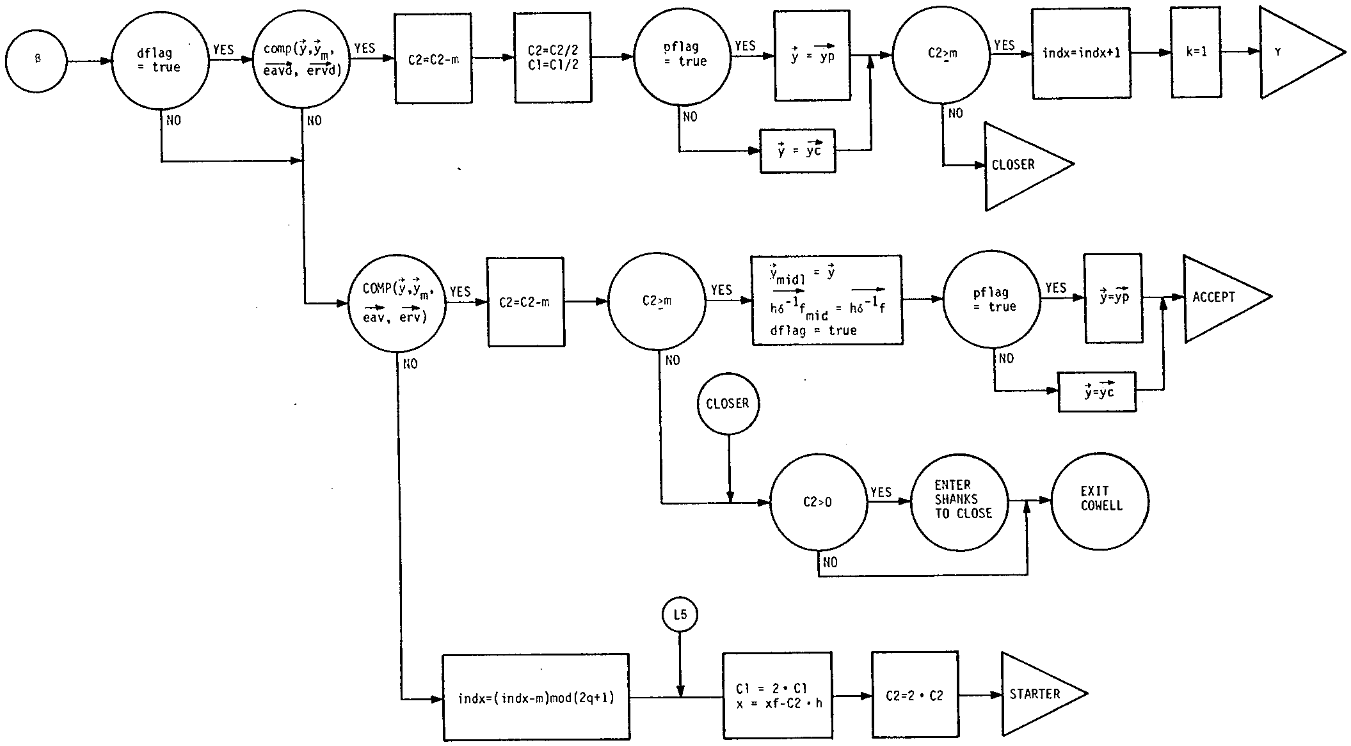
B
dflag = true
YES
NO
comp($\vec{y}$,$\vec{y}_m$, $\overrightarrow{eavd}$, $\overrightarrow{ervd}$)
YES
NO
C2=C2-m
C2=C2/2
C1=C1/2
pflag = true
YES
NO
$\vec{y}$ = $\overrightarrow{yp}$
$\vec{y}$ = $\overrightarrow{yc}$
C2≥m
YES
NO
indx=indx+1
k=1
Y
CLOSER
COMP($\vec{y}$,$\vec{y}_m$, $\overrightarrow{eav}$, $\overrightarrow{erv}$)
YES
NO
C2=C2-m
C2≥m
YES
NO
$\vec{y}_{midl}$ = $\vec{y}$
$\overrightarrow{h\delta^{-1}f_{mid}}$ = $\overrightarrow{h\delta^{-1}f}$
dflag = true
pflag = true
YES
NO
$\vec{y}$=$\overrightarrow{yp}$
$\vec{y}$=$\overrightarrow{yc}$
ACCEPT
CLOSER
C2>0
YES
NO
ENTER SHANKS TO CLOSE
EXIT COWELL
L5
indx=(indx-m)mod(2q+1)
C1 = 2 • C1
x = xf-C2 • h
C2=2 • C2
STARTER

Figure 3 (Continued). Flow Diagram for the Cowell Method.

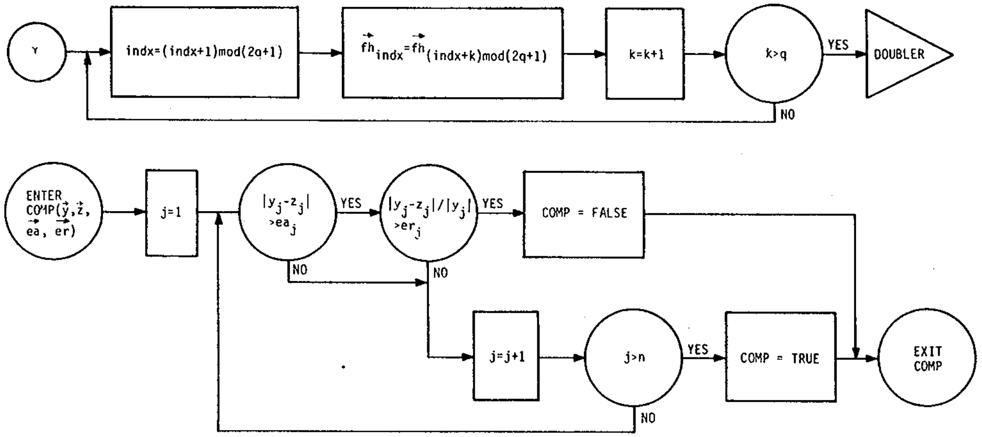
Y
indx=(indx+1)mod(2q+1)
$\vec{fh}_{indx}=\vec{fh}_{(indx+k)mod(2q+1)}$
k=k+1
k>q
YES
NO
DOUBLER
ENTER COMP($\vec{y}$,$\vec{z}$, $\vec{ea}$, $\vec{er}$)
j=1
$|y_j-z_j|$ $>ea_j$
YES
NO
$|y_j-z_j|/|y_j|$ $>er_j$
YES
NO
COMP = FALSE
j=j+1
j>n
YES
NO
COMP = TRUE
EXIT COMP

Figure 3 (Continued). Flow Diagram for the Cowell Method.

and it is the same factor that was determined in earlier experiments with $p = 0$ [18].

One other consequence of the present error control is the limitation of accuracy obtainable with a single precision program. When the relative estimated error is required to be less than 10^{-11} for doubling to occur, doubling is almost precluded since the computer can only carry eleven to twelve decimal digits. Under such conditions the number of steps increases enormously, and the program is virtually useless. For $p = 0$ and variables of order of magnitude one, this situation occurs when the error asked divided by $10 \cdot 2^{q+3}$ is about 10^{-11}; however, with $p \neq 0$, this situation occurs when the step size is such that the error asked divided by $10 \cdot cl^{p} \cdot 2^{q+3}$ is about 10^{-11}. Thus the smallest allowable asked error is reduced as p approaches one.

Since the even integer q is also involved in the calculation of the smallest allowable asked error, it becomes apparent that the smallest allowable asked error can be asked with small q, yet the larger values of q offer their biggest advantage at higher asked accuracies. Results show that best single precision results tend to come from runs with $q = 4$ and $q = 6$ at the asked accuracies which are reasonable for single precision; earlier experiments with double precision asking higher accuracies [18] showed that best results came from higher values of q.

The matching of the order of the start procedure with the order of the Cowell method was somewhat difficult due to the limitation on accuracy asked. The (4,4) Shanks formula seemed to give best results for $q = 4$, at all accuracies and best results for $q = 6$ at larger asked errors; the (5,5) Shanks formula seemed to give best results for $q = 6$ at smaller or asked error. These results were not extensive enough to be conclusive, however.

Corrector convergence can become a problem under two conditions. First, the Runge-Kutta-Shanks formulas can take much larger steps than the Cowell method at lower asked accuracies. The step size chosen by the start procedure can be large enough so that the Cowell corrector will not converge, yet the steps are accurate enough as Runge-Kutta-Shanks steps. Second, in rapidly approaching a singularity the step size could suddenly become too large for Cowell corrector convergence, for step size control is only exercised after each m steps. The corrector counter was required to protect against the corrector not converging; a halving is called for whenever more than eight times through the corrector become necessary.

A final result concerns second order systems. Cowell's method was originally a pair of predictor and corrector formulas to be used to compute the positions as well as the velocities directly from the function value history. The predictor and corrector to compute the positions was of one higher order than the corresponding velocity formulas. Earlier experiments [18] were made using this type of approach. Present experiments required the second order system to be reduced to a first order system; the predictor and corrector are simply the velocity formulas. Both earlier and present experiments show the positions to be more accurate than the velocities; hence only time, not accuracy, is lost when a second order system must be solved as a first order system.

```
PROCEDURE COWELL(N,XI,XF,Y,F,EA,ER,P,DX,RKSFN,RKSORDER,RKSCOEFF,Q,
COWELLCOEFF,START,SHANKS);
VALUE N,XI,XF,P,DX,RKSFN,RKSORDER,Q ;
INTEGER N,RKSFN,RKSORDER,Q ;
REAL XI,XF,P,DX ;
REAL ARRAY Y,EA,ER,RKSCOEFF,COWELLCOEFF[0];
PROCEDURE F,SHANKS ;
INTEGER PROCEDURE START ;

BEGIN
  INTEGER C1,M,MM1,QP1,TQP1,INDX,I1,I2,I3,I,J,K,CYI ;
  INTEGER CORRECTCNT ;
  REAL INT,C2,DFACTOR,X,H,T1,T2,T3,T4,T5,T6 ;
  BOOLEAN DFLAG,PFLAG ;
  REAL ARRAY FH[0:Q +Q,0:N],YMID1,YP,YC,YM,CS,FP,HDM1F,HDM1FMID,EAV,ERV
  ,EAVD,ERVD[0:N],PCOEFF,CCOEFF,MCOEFF[0:Q];
  LABEL L0,L1,L2,L3,L4,L5,STARTER,DOUBLER,ACCEPT,CORRECT,CLOSER;
  INT :=XF -XI ;
  C1 :=1 ;
  L0:C1 :=C1 +C1 ;
  IF C1 <Q THEN GO TO L0 ;
  IF DX <0 THEN DX :=-DX ;
  IF DX ≠0 THEN
  BEGIN
    L1:IF ABS(INT)/C1 >DX THEN
    BEGIN
      C1 :=C1 +C1 ;
      GO TO L1
    END
  END ;
  C2 :=C1 ;
  M :=Q DIV 2 ;
  MM1 :=M -1 ;
  QP1 :=Q +1 ;
  TQP1 :=QP1 +Q ;
  DFACTOR :=2.0 *(Q +3);
  INDX :=0 ;
  X :=XI ;
  F(N,X,Y,FH[0,*]);
  STARTER:C1 :=(I1 :=START(N,XI,XF,C1,EA,ER,F,Q,X,Y,FH,FH,YMID1,INDX,
TQP1,1,P,RKSFN,RKSCOEFF))×C1 ;
C2 :=C2 ×I1 -Q ;
INDX :=(INDX +Q)MOD TQP1 ;
IF C2 <M THEN GO TO CLOSER ;
DOUBLER:H :=INT /C1 ;
FOR K :=0 STEP 1 UNTIL Q DO
BEGIN
  PCOEFF[K]:=COWELLCOEFF[K]×H ;
  CCOEFF[K]:=COWELLCOEFF[I1 :=K +QP1]×H ;
  MCOEFF[K]:=COWELLCOEFF[I1 +QP1]×H
END ;
T1 :=(C1 *P)×10.0 ;
FOR J :=1 STEP 1 UNTIL N DO
BEGIN
  EAVD[J]:=(EAV[J]:=EA[J]/T1)/DFACTOR ;
  ERVD[J]:=(ERV[J]:=ER[J]/T1)/DFACTOR
END ;
T1 :=MCOEFF[0];
FOR J :=1 STEP 1 UNTIL N DO HDM1F[J]:=YMID1[J]-FH[INDX,J]×T1 ;
CYI :=INDX +TQP1 ;
I3 :=CYI -QP1 ;
FOR K :=1 STEP 1 UNTIL M DO
BEGIN
  I1 :=(CYI -K)MOD TQP1 ;
  I2 :=(I3 +K)MOD TQP1 ;
  T1 :=MCOEFF[K]-H ;
  T2 :=MCOEFF[QP1 -K];
  FOR J :=1 STEP 1 UNTIL N DO HDM1F[J]:=HDM1F[J]-FH[I1,J]×T1 -FH[I2,J]
  ×T2
END ;
FOR J :=1 STEP 1 UNTIL N DO HDM1FMID[J]:=HDM1F[J];
DFLAG :=FALSE ;
ACCEPT:FOR I :=1 STEP 1 UNTIL M DO
BEGIN
  CYI :=(INDX :=(INDX +1)MOD TQP1)+TQP1 ;
  X :=XF -(C2 -I)×H ;
  I1 :=(CYI -1)MOD TQP1 ;
  T1 :=PCOEFF[0];
  T2 :=CCOEFF[1];
  FOR J :=1 STEP 1 UNTIL N DO
```

```
BEGIN
  YP[J]:=(T4 :=(HDM1F[J]:=HDM1F[J]+H ×(T3 :=FH[I1,J])))+T1 ×T3 ;
  CS[J]:=T4 +T2 ×T3
END ;
I3 :=CYI -QP1 ;
FOR K :=2 STEP 1 UNTIL M DO
BEGIN
  I1 :=(CYI -K)MOD TQP1 ;
  I2 :=(I3 +K)MOD TQP1 ;
  T1 :=PCOEFF[K -1];
  T2 :=CCOEFF[K];
  T3 :=PCOEFF[Q -K];
  T4 :=CCOEFF[QP1 -K];
  FOR J :=1 STEP 1 UNTIL N DO
  BEGIN
    YP[J]:=YP[J]+T1 ×(T5 :=FH[I1,J])+T3 ×(T6 :=FH[I2,J]);
    CS[J]:=CS[J]+T2 ×T5 +T4 ×T6
  END
END ;
I1 :=(CYI -Q)MOD TQP1 ;
I2 :=(CYI -QP1)MOD TQP1 ;
T1 :=PCOEFF[Q -1];
T2 :=CCOEFF[Q];
T3 :=PCOEFF[Q];
FOR J :=1 STEP 1 UNTIL N DO
BEGIN
  YP[J]:=YP[J]+T1 ×(T4 :=FH[I1,J])+T3 ×FH[I2,J];
  CS[J]:=CS[J]+T2 ×T4
END ;
T2 :=CCOEFF[0];
CORRECTCNT :=1 ;
CORRECT:F(N,X,YP,FP);
FOR J :=1 STEP 1 UNTIL N DO IF (T1 :=ABS((T3 :=(YC[J]:=CS[J]+T2 ×FP
[J]))-YP[J]))>EAV[J]THEN
BEGIN
  IF T1 >ERV[J]×ABS(Y3)THEN
  BEGIN
    FOR J :=J +1 STEP 1 UNTIL N DO YC[J]:=CS[J]+T2 ×FP[J];
    F(N,X,YC,FP);
    FOR J :=1 STEP 1 UNTIL N DO IF (T1 :=ABS((T3 :=(YP[J]:=CS[J]+T2
      ×FP[J]))-YC[J]))>EAV[J]THEN
      BEGIN
        IF T1 >ERV[J]×ABS(T3)THEN
        BEGIN
          FOR J :=J +1 STEP 1 UNTIL N DO YP[J]:=CS[J]+T2 ×FP[J];
          CORRECTCNT :=CORRECTCNT +2 ;
          IF CORRECTCNT >8 THEN
          BEGIN
            INDX :=(CYI -I)MOD TQP1 ;
            GO TO L5
          END ;
          GO TO CORRECT
        END
      END ;
      F(N,X,YP,FH[INDX,*]);
      PFLAG :=TRUE ;
      CORRECTCNT :=CORRECTCNT +1 ;
      GO TO L2
    END
  END ;
  F(N,X,YC,FH[INDX,*]);
  PFLAG :=FALSE ;
  L2:
END ;
I1 :=(CYI -M)MOD TQP1 ;
T1 :=MCOEFF[M];
FOR J :=1 STEP 1 UNTIL N DO YM[J]:=HDM1FMID[J]+T1 ×FH[I1,J];
I3 :=CYI -Q ;
FOR K :=0 STEP 1 UNTIL MM1 DO
BEGIN
  I1 :=(CYI -K)MOD TQP1 ;
  I2 :=(I3 +K)MOD TQP1 ;
  T1 :=MCOEFF[K];
  T2 :=MCOEFF[Q -K];
  FOR J :=1 STEP 1 UNTIL N DO YM[J]:=YM[J]+T1 ×FH[I1,J]+T2 ×FH[I2,J]
END ;
IF DFLAG THEN
BEGIN
  FOR J :=1 STEP 1 UNTIL N DO IF (T2 :=ABS((T3 :=Y[J])-YM[J]))>EAVD[J
  ]THEN
  BEGIN
    IF T2 >ERVD[J]×ABS(T3)THEN
    BEGIN
      IF T2 >EAV[J]THEN
      BEGIN
        IF T2 >ERV[J]×ABS(T3)THEN GO TO L4
      END ;
      GO TO L3
    END
  END ;
  C2 :=C2 -M ;
  C2 :=C2 /2.0 ;
  IF PFLAG THEN FOR J :=1 STEP 1 UNTIL N DO Y[J]:=YP[J]ELSE FOR J :=1
  STEP 1 UNTIL N DO Y[J]:=YC[J];
  C1 :=C1 DIV 2 ;
  IF C2 <M THEN GO TO CLOSER ;
  INDX :=INDX +1 ;
  FOR K :=1 STEP 1 UNTIL Q DO
  BEGIN
    INDX :=(INDX +1)MOD TQP1 ;
    I1 :=(INDX +K)MOD TQP1 ;
    FOR J :=1 STEP 1 UNTIL N DO FH[INDX,J]:=FH[I1,J]
  END ;
  GO TO DOUBLER
END ;
J :=0 ;
L3:FOR J :=J +1 STEP 1 UNTIL N DO IF (T2 :=ABS((T3 :=Y[J])-YM[J]))>EAV
[J]THEN
BEGIN
  IF T2 >ERV[J]×ABS(T3)THEN
  BEGIN
    L4:INDX :=(CYI -M)MOD TQP1 ;
    L5:C1 :=C1 +C1 ;
    X :=XF -C2 ×H ;
    C2 :=C2 +C2 ;
    GO TO STARTER
  END
END ;
C2 :=C2 -M ;
IF C2 ≥M THEN
```

```
  BEGIN5
    IF PFLAG THEN FOR J :=1 STEP 1 UNTIL N DO
    BEGIN
      YMID1[J]:=Y[J];
      Y[J]:=YP[J];
      HDM1FMID[J]:=HDM1F[J]
    END ELSE FOR J :=1 STEP 1 UNTIL N DO
    BEGIN
      YMID1[J]:=Y[J];
      Y[J]:=YC[J];
      HDM1FMID[J]:=HDM1F[J]
    END ;
    DFLAG :=TRUE ;
    GO TO ACCEPT
  END ;
  IF PFLAG THEN FOR J :=1 STEP 1 UNTIL N DO Y[J]:=YP[J]ELSE FOR J :=1
  STEP 1 UNTIL N DO Y[J]:=YC[J];
  CLOSER:IF C2 >0 THEN SHANKS(N,X,XF,Y,F,RKSFN,RKSORDER,RKSCOEFF,P,EA,ER
  ,ABS(INT)/C1);

END ;
```

D. The Runge-Kutta-Shanks Method

1. Introduction

The procedure described is a generalization of the Runge-Kutta method for solving a system of differential equations. It may be applied to an arbitrary system of first-order differential equations of the form

$$\vec{y}' = \vec{f}\,(x,\vec{y})$$

with the initial conditions

$$\vec{y}\,(x_o) = \vec{y}_o$$

where $\vec{y}\,(x) = \begin{pmatrix} y_1\,(x) \\ \vdots \\ y_n\,(x) \end{pmatrix}$, $\qquad \vec{y}'(x) = \begin{pmatrix} y_1'(x) \\ \vdots \\ y_n'(x) \end{pmatrix}$,

$\vec{f}\,(x,\vec{y}) = \begin{pmatrix} f_1\,(x,y_1, \ldots, y_n) \\ \vdots \\ f_n\,(x,y_1, \ldots, y_n) \end{pmatrix}$, $\qquad \vec{y}_o = \begin{pmatrix} y_{10} \\ \vdots \\ y_{n0} \end{pmatrix}$.

2. Description of the Method

The Shanks Method is a single-step procedure for finding a numerical solution of a first-order ordinary differential equation or system of differential equations in which the derivatives of the dependent variables may be expressed explicitly as functions of the independent and dependent variables.

Consider the system of differential equation

$$\vec{y}' = \vec{f}\,(x,\vec{y}) \quad .$$

Suppose the value of $\vec{y}$ (x) is known. The value $\vec{y}$ (x + h) is approximated by

$$\vec{y}\,(x + h) = \vec{y}\,(x) + h \sum_{i=1}^{m} \gamma_i \vec{f}_i\,(x,h,\vec{y}),$$

where

$$\vec{f}_1\,(x,h,\vec{y}) = \vec{f}(x,\vec{y}),$$

$$\vec{f}_i\,(x,h,\vec{y}) = \vec{f}\,(x + \alpha_i h,\ \vec{y} + h \sum_{j=1}^{i-1} \beta_{ij}\vec{f}_j),\ i = 2, \ldots, m.$$

The coefficients α_i (i = 2, . . ., m),

β_{ij} (i = 2, . . ., m; j =1, . . ., i-1), and γ_i(i = 1, . . ., m)

are chosen so as to make the approximation correct to some order. A special case of the Shanks formula is the fourth-order Runge-Kutta formula:

$$\alpha_2 = 1/2,\ \alpha_3 = 1/2,\ \alpha_4 = 1,$$

$$\beta_{21} = 1/2,\ \beta_{31} = 0,\ \beta_{32} = 1/2,\ \beta_{41} = \beta_{42} = 0,\ \beta_{43} = 1,$$

$$\gamma_1 = 1/6,\ \gamma_2 = 1/3,\ \gamma_3 = 1/3,\ \gamma_4 = 1/6.$$

For useful values of the various combinations of α, β, and γ, see Shanks [17].

3. The Computer Procedure

The procedure was programmed for the B-5500 computer in the B-5500 Algol language. Single precision arithmetic (11 to 12 decimal digits) was used.

3.1 Error Estimates and Step Size Control

In this procedure a single set of Shanks formulas is used. Suppose a vector $\vec{y}$ (x) is known. Then the Shanks method is applied to one step of size h(where h = $\frac{\Delta x}{c}$, Δx is the length of the interval, and c is a power of two), and to two steps of size $\frac{h}{2}$, as follows:

$$\vec{y}_p = \vec{y}\ (x) + h \sum_{i=1}^{m} \gamma_i \vec{f}_i(x,h,\vec{y}),$$

$$\vec{y}_m = \vec{y}\ (x) + \frac{h}{2} \sum_{i=1}^{m} \gamma_i \vec{f}_i(x,\frac{h}{2},\vec{y})$$

$$\vec{y}_c = \vec{y}_m + \frac{h}{2} \sum_{i=1}^{m} \gamma_i \vec{f}_i(x + \frac{h}{2},\frac{h}{2},\vec{y}_m).$$

Both $\vec{y}_p$ and $\vec{y}_c$ are estimates of $\vec{y}(x+h)$. An error estimate $E_k = \left|\frac{y_{ck} - y_{pk}}{f}\right|$ (where f is an empirical factor) is calculated for each independent variable y_k. If both $E_k > \frac{E_{ak}}{c^p}$ and $E_k > \frac{E_{rk}|y_{ck}|}{c^p}$ for any dependent variable where E_{ak} is an absolute error estimate, E_{rk} is a relative error estimate, and p is an input parameter, usually 1 or 1/2, then the step is rejected and the step size is halved; otherwise the step is accepted and $\vec{y}_c$ is taken as the vector $\vec{y}(x+h)$. If for every dependent variable, either $E_k > \frac{E_{ak}}{2^{(j+3)}c^p}$ or $E_k > \frac{E_{rk}|y_{ck}|}{2^{(j+3)}c^p}$, where j is the order, then the step size is doubled. If the step size h is larger than the distance to the end of the interval, then that distance is taken as the step size.

3.2 <u>Input and Output of the Procedure</u>

The procedure is called as follows:

SHANKS (N,XI,XF,YV,F,M,ORDER,CF,P,EA,ER,DX);

where the parameters have the following meaning:

N - number of dependent variables;

XI - initial value of the independent variable;

XF - final value of the independent variable;

YV - array of initial values of the dependent variables, based at zero but with the zero element not used;

F - a function evaluation procedure, supplied by the user, called as follows

F(N,X,YV,FV);

where N is the number of dependent variables, X is the value of the independent variable, YV is the array of values of the dependent variables, and FV is the array in which the function values are placed;

M - the number of function evaluations in each application of the Shanks method;

ORDER - the order of the Shanks formulas used;

CF - the array of Shanks coefficients, starting in the zero element arranged as follows: for each i, the corresponding $\alpha_i\beta_{ij}$'s, followed by α_i, with the γ_i's at the end;

P - an exponent (usually 1/2 or 1) used in step size control (1 assuming the errors are additive; 1/2 assuming that they are random);

EA - an array of absolute error asked;

ER - an array of relative error asked;

DX - a recommended starting step size (the actual starting step size will be $\frac{XF - XI}{c}$, where c is the smallest power of 2 for which $\left|\frac{XF - XI}{c}\right| \leq \left|DX\right|$).

The final values of the dependent variable are stored in YV before exiting the procedure.

4. Flow Diagram and Program Listing

Figure 4 is the flow diagram for the Runge-Kutta-Shanks procedure. A listing of the program is given at the end of this section.

5. Results and Conclusions

This procedure was used with systems of differential equations with

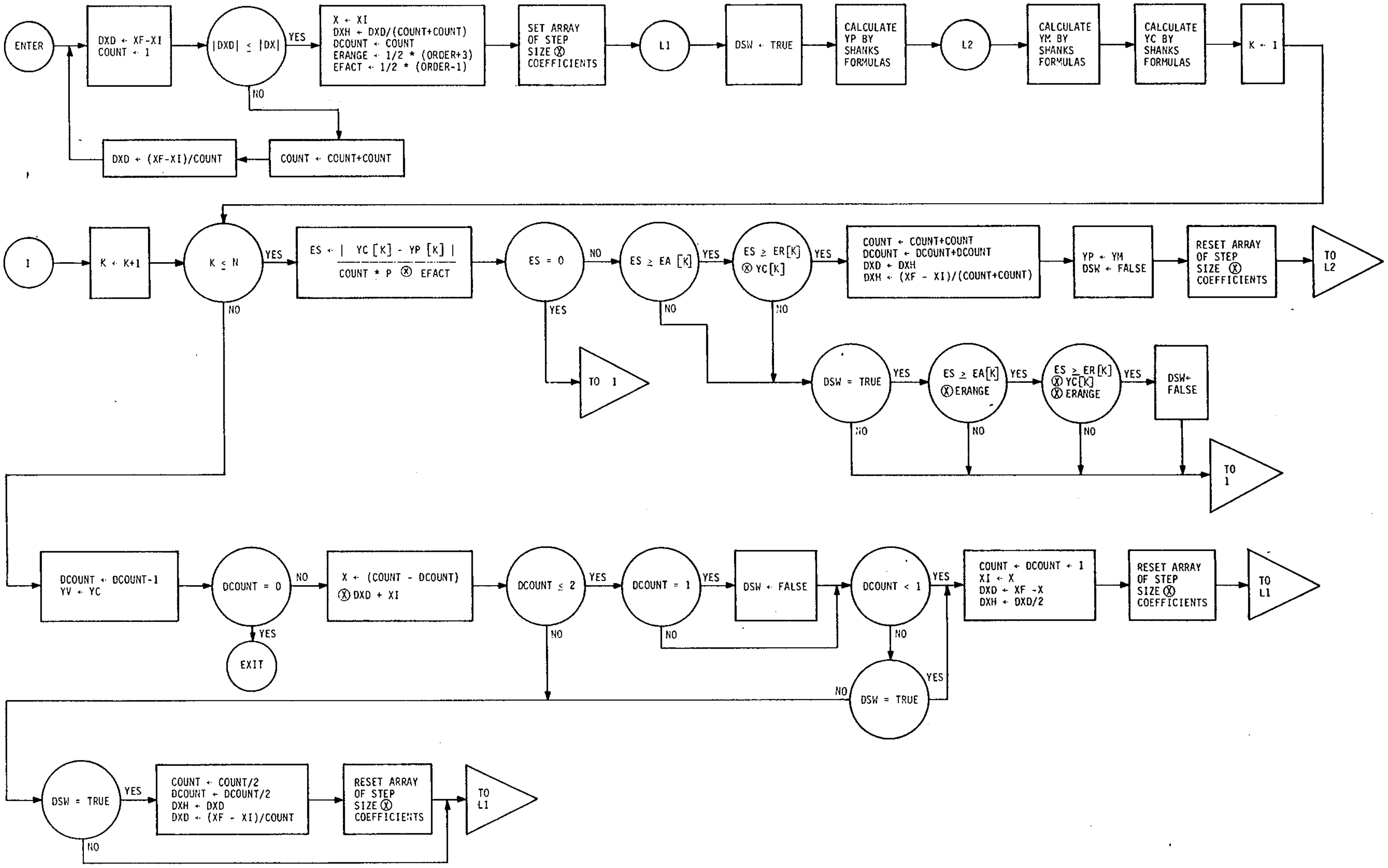

Figure 4. Flow Diagram for the Runge-Kutta-Shanks Procedure.

analytic solutions, as well as with the three-body problem. It gave slightly more accuracy than was asked.

In order to reach the end of the interval more accurately, the steps taken were binary fractions of the total interval. Hence, it was necessary to use halving and doubling rather than the continuous step size control of previous experiments [18]. Although this caused the rejection of more steps, it prevented roundoff in the independent variable.

The procedure was first run without the empirical factor f (i.e., with f = 1) mentioned in 3.1. The results were more accurate than asked.

The theoretical value, $f = \frac{1}{2^{\text{order}}-1}$, was then used. It was found that for some formulas (in particular, 8-10 and 8-12), the desired accuracy was not reached.

Finally, runs were made with a compromise value, $f = \frac{1}{2^{\text{order}-1}}$. In this case, the results were good for most formulas, but the Shanks 8-10 formulas still sometimes did not obtain the desired accuracy.

It might be noted that the most accurate results were usually obtained with the Shanks 4-4 formulas.

It is recommended that further experimentation be done in the area of step-size control with the Runge-Kutta-Shanks method. In particular, other values for the factor f might be used. It might be desirable to determine a particular constant for each set of formulas.

```
PROCEDURE SHANKS(N,XI,XF,YV,F,M,ORDER,CF,P,EA,ER,DX);
VALUE N,XI,XF,M,ORDER,P,DX;
INTEGER N,M,ORDER;
REAL XI,XF,P,DX;
REAL ARRAY YV,CF,EA,ER[0];
PROCEDURE F;

BEGIN
  INTEGER I,J,K,L,COUNT,COUNT2,II,NCF;
  INTEGER NCF1;
  INTEGER DKTR;
  REAL EFACT;
  REAL BETA,DCOUNT,DXD,DXH,DXT,EFACTOR,ERANGE,ES,GAMMA,X,XM;
  BOOLEAN CFSW,DSW;
  REAL ARRAY CFD[0:(M+3)×M-2],FV[0:M-1,0:N],GV,YC,YM,YP[0:N];
  DEFINE CFH=CFD#;
  LABEL L1,L2,EXIT;
  INTEGER STEPR,STEPS;
  M←M-1;
  STEPS←STEPR←0;
  DXD←DXT←XF-XI;
  IF DXT=0THEN GO TO EXIT;
  IF DX=0THEN DX←DXD;
  COUNT←1;
  WHILE ABS(DX)<ABS(DXD)DO
  BEGIN
    COUNT←COUNT+COUNT;
    DXD←DXT/COUNT;

  END;
  COUNT2←COUNT+COUNT;
  DXH←DXT/COUNT2;
  DCOUNT←COUNT;
  EFACT←1;
  FOR I←1STEP 1UNTIL ORDER DO EFACT←EFACT+EFACT;
  ERANGE←0.125/EFACT;
  EFACT←4/EFACT;
  EFACTOR←COUNT*P×EFACT;
  DKTR←0;
  NCF1←(M×M+M)DIV 2+M+M;
```

```
NCF←NCF1+1;
CFSW←FALSE;
FOR I←0STEP 1UNTIL NCF1 DO
BEGIN
  CFD[I]←CF[I]×DXD;
  CFH[I+NCF]←CF[I]×DXH;

END;
X←XI;
XM←XI+DXH;
L1:DSW←TRUE;
F(N,X,YV,GV);
IF CFSW THEN L←NCF1 ELSE L←-1;
FOR I←1STEP 1UNTIL M DO
BEGIN
  II←I-1;
  L←L+1;
  BETA←CFD[L];
  FOR K←1STEP 1UNTIL N DO YP[K]←GV[K]×BETA+YV[K];
  FOR J←1STEP 1UNTIL II DO
  BEGIN
    L←L+1;
    BETA←CFD[L];
    FOR K←1STEP 1UNTIL N DO YP[K]←FV[J,K]×BETA+YP[K];

  END;
  L←L+1;
  F(N,CFD[L]+X,YP,FV[I,*]);

END;
L←L+1;
GAMMA←CFD[L];
FOR K←1STEP 1UNTIL N DO YP[K]←GV[K]×GAMMA+YV[K];
FOR I←1STEP 1UNTIL M DO
BEGIN
  L←L+1;
  GAMMA←CFD[L];
  FOR K←1STEP 1UNTIL N DO YP[K]←FV[I,K]×GAMMA+YP[K];

END;
```

```
L2:IF CFSW THEN L←-1;
FOR I←1STEP 1UNTIL M DO
BEGIN
  II←I-1;
  L←L+1;
  BETA←CFH[L];
  FOR K←1STEP 1UNTIL N DO YM[K]←GV[K]×BETA+YV[K];
  FOR J←1STEP 1UNTIL II DO
  BEGIN
    L←L+1;
    BETA←CFH[L];
    FOR K←1STEP 1UNTIL N DO YM[K]←FV[J,K]×BETA+YM[K];

  END;
  L←L+1;
  F(N,CFH[L]+X,YM,FV[I,*]);

END;
L←L+1;
GAMMA←CFH[L];
FOR K←1STEP 1UNTIL N DO YM[K]←GV[K]×GAMMA+YV[K];
FOR I←1STEP 1UNTIL M DO
BEGIN
  L←L+1;
  GAMMA←CFH[L];
  FOR K←1STEP 1UNTIL N DO YM[K]←FV[I,K]×GAMMA+YM[K];

END;
F(N,XM,YM,FV[0,*]);
IF CFSW THEN L←-1ELSE L←NCF1;
FOR I←1STEP 1UNTIL M DO
BEGIN
  II←I-1;
  FOR K←1STEP 1UNTIL N DO YC[K]←YM[K];
  FOR J←0STEP 1UNTIL II DO
  BEGIN
    L←L+1;
    BETA←CFH[L];
    FOR K←1STEP 1UNTIL N DO YC[K]←FV[J,K]×BETA+YC[K];

  END;
  L←L+1;
  F(N,CFH[L]+XM,YC,FV[I,*]);

END;
FOR K←1STEP 1UNTIL N DO YC[K]←YM[K];
FOR I←0STEP 1UNTIL M DO
BEGIN
  L←L+1;
  GAMMA←CFH[L];
  FOR K←1STEP 1UNTIL N DO YC[K]←FV[I,K]×GAMMA+YC[K];

END;
FOR K←1STEP 1UNTIL N DO
BEGIN
  ES←ABS(YC[K]-YP[K])×EFACTOR;
  IF ES≠0THEN
  BEGIN
    IF ES≥EA[K]THEN IF ES≥ABS(YC[K])×ER[K]THEN
    BEGIN
      DSW←FALSE;
      STEPR←STEPR+1;
      COUNT←COUNT2;
      COUNT2←COUNT+COUNT;
      DCOUNT←DCOUNT+DCOUNT;
      DXD←DXH;
      DXH←DXT/COUNT2;
      EFACTOR←COUNT*P×EFACT;
      IF CFSW THEN
      BEGIN
        FOR I←0STEP 1UNTIL NCF1 DO CFH[I+NCF]←CF[I]×DXH;
        CFSW←FALSE;

      END ELSE
      BEGIN
        FOR I←0STEP 1UNTIL NCF1 DO CFH[I]←CF[I]×DXH;
        CFSW←TRUE;

      END;
      XM←(COUNT2+1-DCOUNT-DCOUNT)×DXH+XI;
```

```
   FOR K←1STEP 1UNTIL N DO YP[K]←YM[K];
   GO TO L2;

  END;
  IF DSW THEN IF ES≥EA[K]×ERANGE THEN IF ES≥ABS(YC[K])×ER[K]×ERANGE
  THEN DSW←FALSE;

 END;

END;
DCOUNT←DCOUNT-1;
STEPS←STEPS+1;
FOR K←1STEP 1UNTIL N DO YV[K]←YC[K];
IF DCOUNT=0THEN GO TO EXIT;
X←(COUNT-DCOUNT)×DXD+XI;
IF DCOUNT<2THEN
BEGIN
 IF DCOUNT=1THEN DSW←FALSE;
 IF DCOUNT<1OR DSW THEN
 BEGIN
  IF DCOUNT>1THEN DKTR←DKTR+1;
  COUNT←DCOUNT+1;
  COUNT2←2;
  EFACTOR←EFACT;
  XI←X;
  DXD←DXT←XF-XI;
  DXH←DXD/2;
  XM←XI+DXH;
  CFSW←FALSE;
  FOR I←0STEP 1UNTIL NCF1 DO
  BEGIN
   CFD[I]←CF[I]×DXD;
   CFH[I+NCF]←CF[I]×DXH;

  END;
  GO TO L1;

 END;

END;
```

```
  IF DSW THEN
  BEGIN
    DKTR←DKTR+1;
    COUNT2←COUNT;
    COUNT←COUNT DIV 2;
    DCOUNT←DCOUNT/2;
    DXH←DXD;
    DXD←DXT/COUNT;
    EFACTOR←COUNT*P×EFACT;
    IF CFSW THEN
    BEGIN
      FOR I←0STEP 1UNTIL NCF1 DO CFD[I]←CF[I]×DXD;
      CFSW←FALSE;

    END ELSE
    BEGIN
      FOR I←0STEP 1UNTIL NCF1 DO CFD[I+NCF]←CF[I]×DXD;
      CFSW←TRUE;

    END;

  END;
  XM←(COUNT2+1-DCOUNT-DCOUNT)×DXH+XI;
  GO TO L1;
  EXIT:
END;
```

E. The General Multistep Method Starting Procedure

1. Introduction

The general multistep method starting procedure is a B-5500 ALGOL single-precision Runge-Kutta-Shanks procedure used for obtaining starting values for the Adams, Butcher, and Cowell multistep methods. The declaration is as follows:

```
integer procedure start (m, xi, xf, cl, ea, er, f, m, x, yiv,
                         yh, fh, yfv, cyi, cym, pa, p,
                         fneval, rksconst);
value n, xi, xf, cl, m, cyi, cym, per, p, fneval;
integer n, cl, m, cyi, cym, pa, fneval
real xi, xf, x, p;
real array ea, er, yiv, yfv, rksconst [0], yh, fh [0,0];
procedure f;
```

2. Description of the Procedure

The parameters of the procedure are defined as follows:

n - the number of dependent variables

xi - the starting value of the independent variable x passed to the multistep method

xf - the final value of the independent variable x passed to the multistep method

cl - the integer counter (xf - xi)/h from the multistep method

ea - the absolute error vector passed to the multistep method

er - the relative error vector passed to the multistep method

f - the procedure which computes $\vec{f}(x,\vec{y}) = \vec{y}'$

m - the number of history points to be calculated by start x - the value of the independent variable at which start begins its integration

yiv - the array which contains on entry for Adams and Cowell the values of the dependent variables at x and which contains on exit for Cowell the values of the dependent variable at the mth point calculated by start

yh - the array which contains on entry for Butcher in row cyi the values of the dependent variables at x and which contains on exit for Butcher the values of the dependent variables at each of the m points calculated by start

fh - the array which contains on entry in row cyi the function values at x and which contains on exit the function values at each of the m points calculated by start

yfv - the array which contains on exit the values of the dependent variables at the mth point calculated by start for Adams or the m/2th point calculated by start for Cowell

cyi - the cyclic index identifying on entry the row of yh in which the values of the dependent variables at x are stored for Butcher and the row of fh in which the function values at x are stored for any method

cym - the number of rows in the arrays yh and fh

pa - the parameter which is zero for Adams, one for Cowell, two for Butcher

p - the exponent such that the absolute error at each step is not to exceed ea/cl^{p} and the relative error at each step is not to exceed er/cl^{p}

fneval - the number of function evaluations required by the Runge-Kutta-Shanks procedure rksconst - the array which contains the Runge-Kutta-Shanks coefficients in the same order as required by the procedure shanks described in section D.

The value of start on exit is two to the power of the number of halvings which took place within start.

Although the base of the arrays ea, er, yiv, and yfv and of the rows of yh and fh is zero, the n components are placed in position 1, 2, ..., n and the zero position is unused.

The procedure attempts to calculate m (if m is even and positive) or m + 1 (if m is odd) Runge-Kutta-Shanks steps of size h = (xf - xi)/cl. After each even step of size h is taken, one step of size 2h is taken over the interval spanned by the two steps of size h. The absolute value of the differences in each dependent variable between the 2h-step and the second h-step is compared with the corresponding component of $\vec{ea}/(cl/2)^p$ for absolute error and with the product of the corresponding component of $\vec{er}/(cl/2)^p$ and the corresponding dependent variable value from the second h-step for relative error. If each component of the difference does not exceed in either the absolute or the relative error test and m steps have not yet been taken, the process of two h-steps, one 2h-step, and test is continued. If any component of the difference exceeds in both the absolute and the relative error tests, cl is doubled, h is halved, and integration begins again at x. The first step of previous size h was saved and becomes the first step of present size 2h.

The m calculated function values from h-steps are placed in rows (cyi+1) mod cym, (cyi+2) mod cym, ..., (cyi+m) mod cym of the array fh. For Butcher, the corresponding dependent variable values from h-steps are placed in the corresponding rows of the array yh; if m is odd, the values of the dependent variable after h-step m + 1 are placed in row (cyi + m + 1) mod cym of yh. For Adams, the dependent variable values from h-step m are placed in the array yfv. For Cowell, the dependent variable values from h-step m are placed in the array yiv and from h-step m/2 (m is always even for Cowell) are placed in yfv. If m is zero, no calculation takes place.

3. Flow Diagram and Program Listing

Figure 5 is the flow diagram for the starting procedure. The program listing follows at the end of this section.

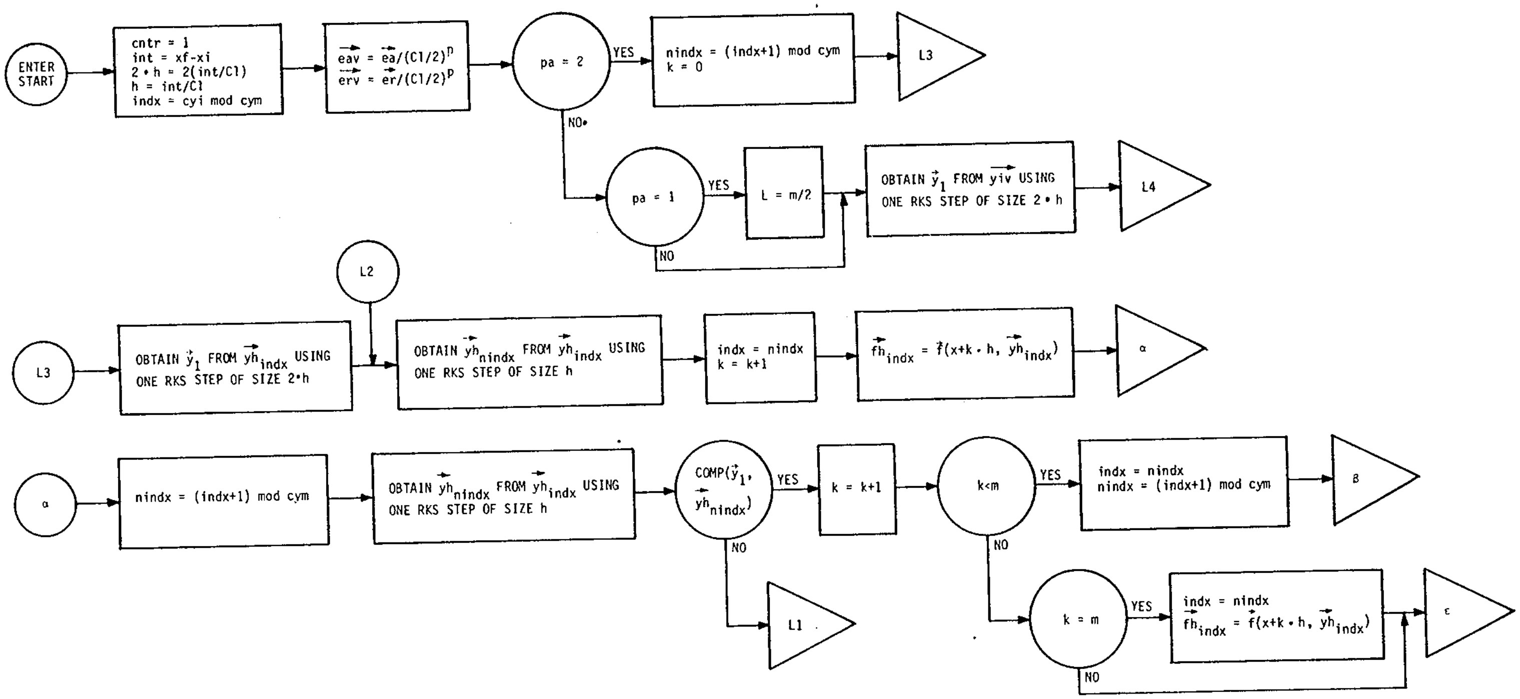
ENTER START
cntr = 1
int = xf-xi
2 • h = 2(int/C1)
h = int/C1
indx = cyi mod cym
$\vec{eav} = \vec{ea}/(C1/2)^p$
$\vec{erv} = \vec{er}/(C1/2)^p$
pa = 2
YES
nindx = (indx+1) mod cym
k = 0
L3
NO•
pa = 1
YES
L = m/2
NO
OBTAIN $\vec{y}_1$ FROM $\vec{yiv}$ USING ONE RKS STEP OF SIZE 2 • h
L4
L2
L3
OBTAIN $\vec{y}_1$ FROM $\vec{yh}_{indx}$ USING ONE RKS STEP OF SIZE 2•h
OBTAIN $\vec{yh}_{nindx}$ FROM $\vec{yh}_{indx}$ USING ONE RKS STEP OF SIZE h
indx = nindx
k = k+1
$\vec{fh}_{indx} = \vec{f}(x+k \cdot h, \vec{yh}_{indx})$
α
α
nindx = (indx+1) mod cym
OBTAIN $\vec{yh}_{nindx}$ FROM $\vec{yh}_{indx}$ USING ONE RKS STEP OF SIZE h
COMP($\vec{y}_1$, $\vec{yh}_{nindx}$)
YES
NO
L1
k = k+1
k<m
YES
indx = nindx
nindx = (indx+1) mod cym
β
NO
k = m
YES
indx = nindx
$\vec{fh}_{indx} = \vec{f}(x+k \cdot h, \vec{yh}_{indx})$
NO
ε

Figure 5. Flow Diagram for the General Multistep Method Starting Procedure.

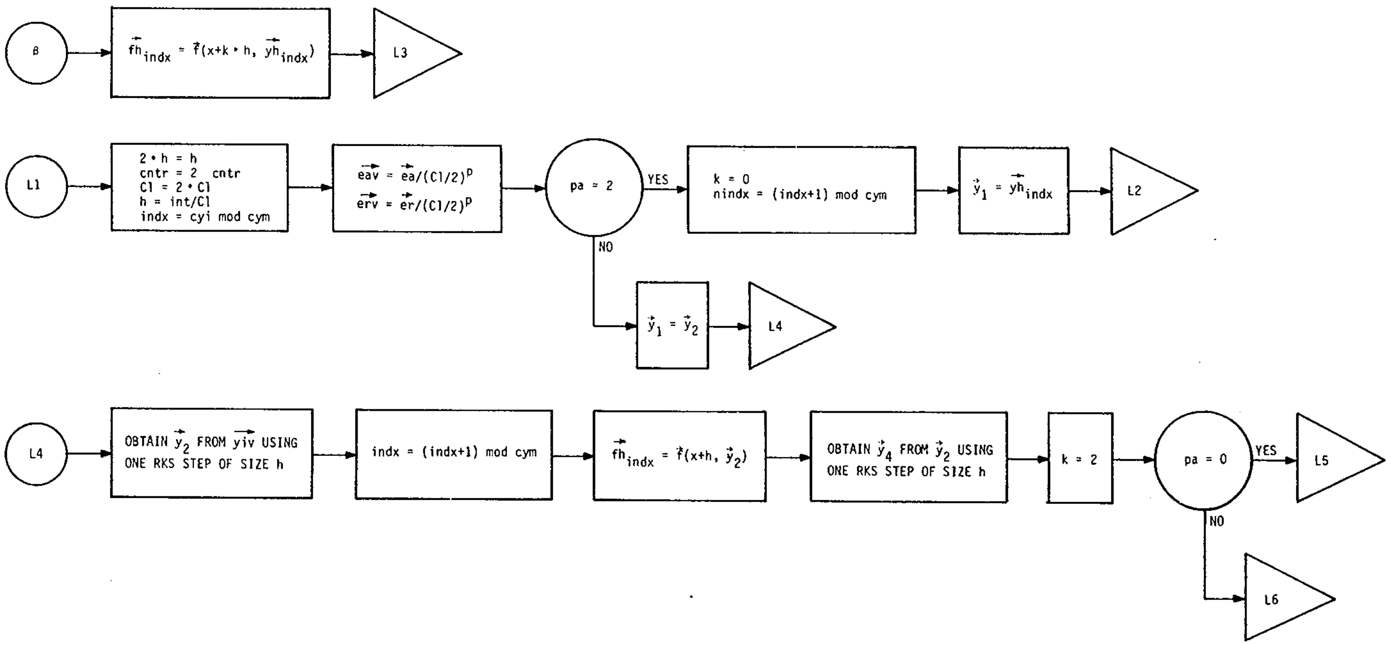
β
$\vec{fh}_{indx} = \vec{f}(x+k \cdot h, \vec{yh}_{indx})$
L3
L1
$2 \cdot h = h$
cntr = 2 cntr
$Cl = 2 \cdot Cl$
h = int/Cl
indx = cyi mod cym
$\vec{eav} = \vec{ea}/(Cl/2)^p$
$\vec{erv} = \vec{er}/(Cl/2)^p$
pa = 2
YES
k = 0
nindx = (indx+1) mod cym
$\vec{y}_1 = \vec{yh}_{indx}$
L2
NO
$\vec{y}_1 = \vec{y}_2$
L4
L4
OBTAIN $\vec{y}_2$ FROM $\vec{yiv}$ USING ONE RKS STEP OF SIZE h
indx = (indx+1) mod cym
$\vec{fh}_{indx} = \vec{f}(x+h, \vec{y}_2)$
OBTAIN $\vec{y}_4$ FROM $\vec{y}_2$ USING ONE RKS STEP OF SIZE h
k = 2
pa = 0
YES
L5
NO
L6

Figure 5 (Continued). Flow Diagram for the General Multistep Method Starting Procedure.

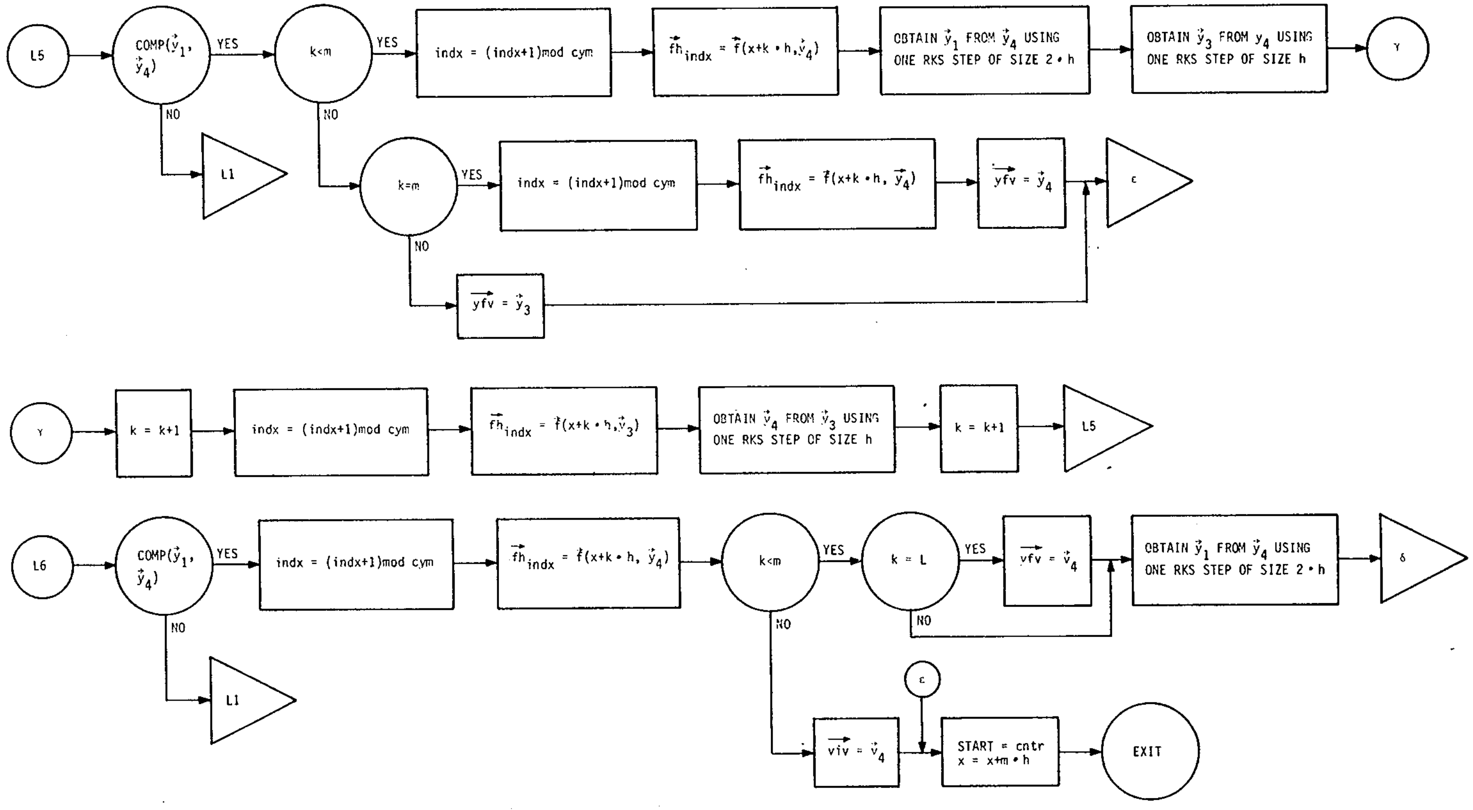
L5
COMP($\vec{y}_1$, $\vec{y}_4$)
YES
NO
L1
k<m
YES
NO
indx = (indx+1)mod cym
$\vec{fh}_{indx} = \vec{f}(x+k \cdot h, \vec{y}_4)$
OBTAIN $\vec{y}_1$ FROM $\vec{y}_4$ USING ONE RKS STEP OF SIZE 2 • h
OBTAIN $\vec{y}_3$ FROM y_4 USING ONE RKS STEP OF SIZE h
γ
k=m
YES
NO
indx = (indx+1)mod cym
$\vec{fh}_{indx} = \vec{f}(x+k \cdot h, \vec{y}_4)$
$\overrightarrow{yfv} = \vec{y}_4$
ε
$\overrightarrow{yfv} = \vec{y}_3$
γ
k = k+1
indx = (indx+1)mod cym
$\vec{fh}_{indx} = \vec{f}(x+k \cdot h, \vec{y}_3)$
OBTAIN $\vec{y}_4$ FROM $\vec{y}_3$ USING ONE RKS STEP OF SIZE h
k = k+1
L5
L6
COMP($\vec{y}_1$, $\vec{y}_4$)
YES
NO
L1
indx = (indx+1)mod cym
$\vec{fh}_{indx} = \vec{f}(x+k \cdot h, \vec{y}_4)$
k<m
YES
NO
k = L
YES
NO
$\overrightarrow{yfv} = \vec{v}_4$
OBTAIN $\vec{y}_1$ FROM $\vec{y}_4$ USING ONE RKS STEP OF SIZE 2 • h
δ
$\overrightarrow{viv} = \vec{v}_4$
ε
START = cntr
x = x+m • h
EXIT

Figure 5 (Continued). Flow Diagram for the General Multistep Method Starting Procedure.

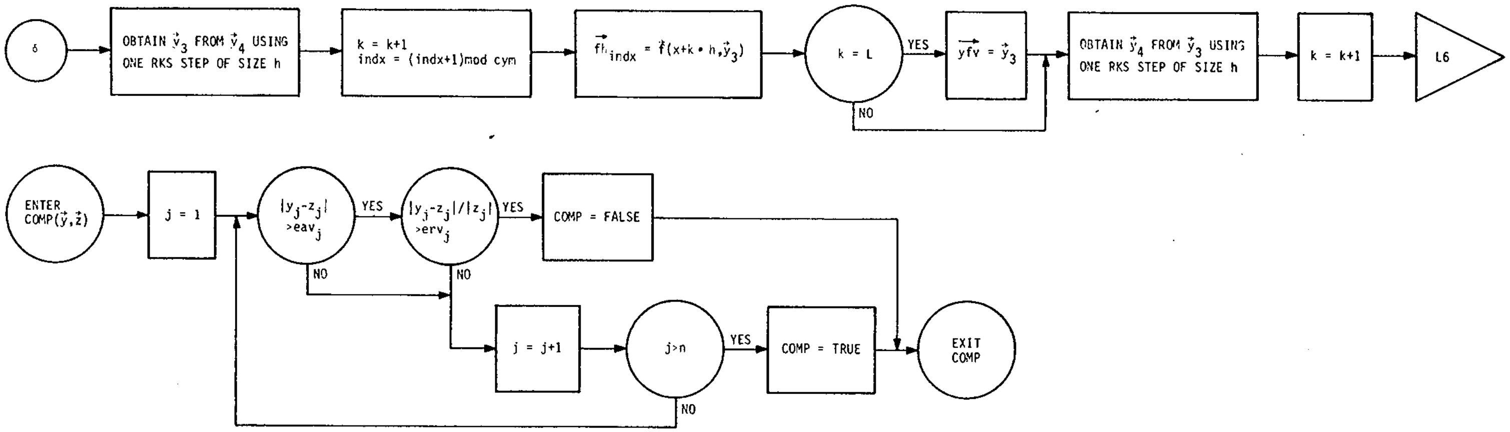
6
OBTAIN $\vec{y}_3$ FROM $\vec{y}_4$ USING ONE RKS STEP OF SIZE h
k = k+1
indx = (indx+1)mod cym
$\overrightarrow{fh}_{indx} = \vec{f}(x+k\cdot h,\vec{y}_3)$
k = L
YES
NO
$\overrightarrow{yfv} = \vec{y}_3$
OBTAIN $\vec{y}_4$ FROM $\vec{y}_3$ USING ONE RKS STEP OF SIZE h
k = k+1
L6
ENTER COMP($\vec{y},\vec{z}$)
j = 1
$|y_j-z_j| > eav_j$
YES
NO
$|y_j-z_j|/|z_j| > erv_j$
YES
NO
COMP = FALSE
j = j+1
j>n
YES
NO
COMP = TRUE
EXIT COMP

Figure 5 (Continued). Flow Diagram for the General Multistep Method Starting Procedure.

```
INTEGER PROCEDURE START(N,XI,XF,C1,EA,ER,F,M,X,YIV,YH,FH,YFV,CYI,CYM,PA 00000000
,P,FNEVAL,RKSCONST);                                                     00001000
VALUE N,XI,XF,C1,M,CYI,CYM,PA,P,FNEVAL ;                                 00002000
INTEGER N,C1,M,CYI,CYM,PA,FNEVAL ;                                       00003000
REAL XI,XF,X,P ;                                                         00004000
REAL ARRAY EA,ER,YIV,YFV,RKSCONST[0],YH,FH[0,0];                         00005000
PROCEDURE F ;                                                            00006000
                                                                         00007000
BEGIN                                                                    00008000
  INTEGER I,J,K,L,COEFFCNT,FNMAX,INDX,NINDX,CNTR ;                       00009000
  REAL INT,H,TWOH,T1 ;                                                   00010000
  REAL ARRAY HC,TWOHC[0:(((FNEVAL+3)×FNEVAL)/2)-2],EAV,ERV,Y1,Y2,Y3,Y4[000011000
  :N],G[0:FNEVAL -2,0:N];                                                00012000
  LABEL L1,L2,L3,L4,L5,L6 ;                                              00013000
  PROCEDURE RUNKUT(N,X,FNMAX,COEFF,YIV,YFV,FV,F,G);                      00014000
  VALUE N,X,FNMAX ;                                                      00015000
  INTEGER N,FNMAX ;                                                      00016000
  REAL X ;                                                               00017000
  REAL ARRAY COEFF,YIV,YFV,FV[0],G[0,0];                                 00018000
  PROCEDURE F ;                                                          00019000
                                                                         00020000
  BEGIN                                                                  00021000
    INTEGER I,J,K,CCNT ;                                                 00022000
    REAL TEMP ;                                                          00023000
    TEMP :=COEFF[CCNT :=0];                                              00024000
    FOR I :=0 STEP 1 UNTIL FNMAX DO                                      00025000
    BEGIN                                                                00026000
      FOR J :=1 STEP 1 UNTIL N DO YFV[J]:=FV[J]×TEMP +YIV[J];            00027000
      FOR K :=0 STEP 1 UNTIL I -1 DO                                     00028000
      BEGIN                                                              00029000
        TEMP :=COEFF[CCNT :=CCNT +1];                                    00030000
        FOR J :=1 STEP 1 UNTIL N DO YFV[J]:=G[K,J]×TEMP +YFV[J]          00031000
      END ;                                                              00032000
      F(N,X +COEFF[CCNT :=CCNT +1],YFV,G[I,*]);                          00033000
      TEMP :=COEFF[CCNT :=CCNT +1]                                       00034000
    END ;                                                                00035000
    FOR J :=1 STEP 1 UNTIL N DO YFV[J]:=FV[J]×TEMP +YIV[J];              00036000
    FOR K :=0 STEP 1 UNTIL FNMAX DO                                      00037000
    BEGIN                                                                00038000
      TEMP :=COEFF[CCNT :=CCNT +1];                                      00039000
    FOR J :=1 STEP 1 UNTIL N DO YFV[J]:=G[K,J]×TEMP +YFV[J]
  END ;

END ;
BOOLEAN PROCEDURE COMP(N,EAV,ERV,Y,Z);
VALUE N ;
INTEGER N ;
REAL ARRAY EAV,ERV,Y,Z[0];

BEGIN
  INTEGER J ;
  REAL T1 ;
  LABEL L1 ;
  FOR J :=1 STEP 1 UNTIL N DO IF (T1 :=ABS(Y[J]-Z[J]))>EAV[J]THEN
  BEGIN
    IF T1 >ERV[J]×ABS(Z[J])THEN
    BEGIN
      COMP :=FALSE ;
      GO TO L1
    END
  END ;
  COMP :=TRUE ;
  L1:
END ;
CNTR :=1 ;
IF M ≠0 THEN
BEGIN
  COEFFCNT :=(((FNEVAL +3)×FNEVAL)/2)-2 ;
  FNMAX :=FNEVAL -2 ;
  INT :=XF -XI ;
  TWOH :=(INT +INT)/C1 ;
  H :=INT /C1 ;
  FOR I :=0 STEP 1 UNTIL COEFFCNT DO
  BEGIN
    HC[I]:=(T1 :=RKSCONST[I])×H ;
    TWOHC[I]:=T1 ×TWOH
  END ;
  INDX :=CYI MOD CYM ;
  T1 :=(C1 /2)*P ;
  FOR J :=1 STEP 1 UNTIL N DO
```

```
BEGIN                                                                   00080000
  EAV[J]:=EA[J]/T1 ;                                                    00081000
  ERV[J]:=ER[J]/T1                                                      00082000
END ;                                                                   00083000
IF PA =2 THEN                                                           00084000
BEGIN                                                                   00085000
  NINDX :=(INDX +1)MOD CYM ;                                            00086000
  K :=0 ;                                                               00087000
  GO TO L3                                                              00088000
END ;                                                                   00089000
IF PA =1 THEN L :=M DIV 2 ;                                             00090000
RUNKUT(N,X,FNMAX,TWOHC,YIV,Y1,FH[INDX,*],F,G);                          00091000
GO TO L4 ;                                                              00092000
L1:TWOH :=H ;                                                           00093000
CNTR :=CNTR +CNTR ;                                                     00094000
C1 :=C1 +C1 ;                                                           00095000
H :=INT /C1 ;                                                           00096000
FOR I :=0 STEP 1 UNTIL COEFFCNT DO                                      00097000
BEGIN                                                                   00098000
  TWOHC[I]:=HC[I];                                                      00099000
  HC[I]:=RKSCONST[I]×H                                                  00100000
END ;                                                                   00101000
INDX :=CYI MOD CYM ;                                                    00102000
T1 :=(C1 /2)*P ;                                                        00103000
IF PA =2 THEN                                                           00104000
BEGIN                                                                   00105000
  K :=0 ;                                                               00106000
  NINDX :=(INDX +1)MOD CYM ;                                            00107000
  FOR J :=1 STEP 1 UNTIL N DO                                           00108000
  BEGIN                                                                 00109000
    EAV[J]:=EA[J]/T1 ;                                                  00110000
    ERV[J]:=ER[J]/T1 ;                                                  00111000
    Y1[J]:=YH[NINDX,J]                                                  00112000
  END ;                                                                 00113000
  L2:RUNKUT(N,K ×H +X,FNMAX,HC,YH[INDX,*],YH[NINDX,*],FH[INDX,*],F,G00114000
  );                                                                    00115000
  K :=K +1 ;                                                            00116000
  INDX :=NINDX ;                                                        00117000
  NINDX :=(INDX +1)MOD CYM ;                                            00118000
  F(N,K ×H +X,YH[INDX,*],FH[INDX,*]);                                   00119000
  RUNKUT(N,K ×H +X,FNMAX,HC,YH[INDX,*],YH[NINDX,*],FH[INDX,*],F,G);
  IF COMP(N,EAV,ERV,Y1,YH[NINDX,*])THEN
  BEGIN
    K :=K +1 ;
    IF K <M THEN
    BEGIN
      INDX :=NINDX ;
      NINDX :=(INDX +1)MOD CYM ;
      F(N,K ×H +X,YH[INDX,*],FH[INDX,*]);
      L3:RUNKUT(N,K ×H +X,FNMAX,TWOHC,YH[INDX,*],Y1,FH[INDX,*],F,G);
      GO TO L2
    END ;
    IF K =M THEN
    BEGIN
      INDX :=NINDX ;
      F(N,K ×H +X,YH[INDX,*],FH[INDX,*])
    END
  END ELSE GO TO L1
END ELSE
BEGIN
  FOR J :=1 STEP 1 UNTIL N DO
  BEGIN
    EAV[J]:=EA[J]/T1 ;
    ERV[J]:=ER[J]/T1 ;
    Y1[J]:=Y2[J]
  END ;
  L4:RUNKUT(N,X,FNMAX,HC,YIV,Y2,FH[INDX,*],F,G);
  INDX :=(INDX +1)MOD CYM ;
  F(N,X +H,Y2,FH[INDX,*]);
  RUNKUT(N,X +H,FNMAX,HC,Y2,Y4,FH[INDX,*],F,G);
  K :=2 ;
  IF PA =0 THEN
  BEGIN
    L5:IF COMP(N,EAV,ERV,Y1,Y4)THEN
    BEGIN
      IF K <M THEN
      BEGIN
        INDX :=(INDX +1)MOD CYM ;
        F(N,K ×H +X,Y4,FH[INDX,*]);
        RUNKUT(N,K ×H +X,FNMAX,TWOHC,Y4,Y1,FH[INDX,*],F,G);
```

```
          RUNKUT(N,K ×H +X,FNMAX,HC,Y4,Y3,FH[INDX,*],F,G);
          K :=K +1 ;
          INDX :=(INDX +1)MOD CYM ;
          F(N,K ×H +X,Y3,FH[INDX,*]);
          RUNKUT(N,K ×H +X,FNMAX,HC,Y3,Y4,FH[INDX,*],F,G);
          K :=K +1 ;
          GO TO L5
        END ;
        IF K =M THEN
        BEGIN
          INDX :=(INDX +1)MOD CYM ;
          F(N,K ×H +X,Y4,FH[INDX,*]);
          FOR J :=1 STEP 1 UNTIL N DO YFV[J]:=Y4[J]
        END ELSE FOR J :=1 STEP 1 UNTIL N DO YFV[J]:=Y3[J]
      END ELSE GO TO L1
    END ELSE
    BEGIN
      L6:IF COMP(N,EAV,ERV,Y1,Y4)THEN
      BEGIN
        INDX :=(INDX +1)MOD CYM ;
        F(N,K ×H +X,Y4,FH[INDX,*]);
        IF K <M THEN
        BEGIN
          IF K =L THEN FOR J :=1 STEP 1 UNTIL N DO YFV[J]:=Y4[J];
          RUNKUT(N,K ×H +X,FNMAX,TWOHC,Y4,Y1,FH[INDX,*],F,G);
          RUNKUT(N,K ×H +X,FNMAX,HC,Y4,Y3,FH[INDX,*],F,G);
          K :=K +1 ;
          INDX :=(INDX +1)MOD CYM ;
          F(N,K ×H +X,Y3,FH[INDX,*]);
          IF K =L THEN FOR J :=1 STEP 1 UNTIL N DO YFV[J]:=Y3[J];
          RUNKUT(N,K ×H +X,FNMAX,HC,Y3,Y4,FH[INDX,*],F,G);
          K :=K +1 ;
          GO TO L6
        END ;
        FOR J :=1 STEP 1 UNTIL N DO YIV[J]:=Y4[J]
      END ELSE GO TO L1
    END
  END ;
  X :=M ×H +X
END ;
```

```
    START :=CNTR ;                                                     00200000
                                                                       00201000
  END ;                                                                00202000
```

III. THE EXECUTIVE PROCEDURE

A. Introduction

The executive procedure acts in an administrative and supervisory capacity. It does the bookkeeping and makes the decisions as to which methods are to be used, but does none of the actual integration. The executive procedure uses as subprocedures five basic integration routines; these are:

1) The Adams-Bashforth-Moulton routine,

2) The Stetter-Gragg-Butcher routine,

3) The Cowell constant Nth order difference routine,

4) The Runge-Kutta-Shanks routine,

5) The start and restart routine
(containing a separate Runge-Kutta-Shanks routine).

These five basic routines do the actual integration. Each is described in Chapter II of this report.

The executive procedure works in the following way. When a call is made in the procedure to integrate from point a to the point b, this interval is divided into eighths. The first eighth of the interval is integrated by one method for each of two different orders, and the time taken by each recorded. The second eighth is integrated by another method, also for two different orders, and the times recorded. The winners then compete against each other over the next fourth of the interval. That is, the fastest order of the first method and the faster order of the second method are both used to integrate the second fourth of the interval, and the time taken by each recorded. The faster method of these two is then presumably the best (fastest) of the four tried, and it is used (alone) to integrate over the last half of the interval.

All of the times measured above are then logged in a cumulative history file and the winners and losers noted.

This file then is used as the basis of selecting which methods and orders are chosen each time. The selection process is as follows: The first of the two methods is chosen at random. The second method is chosen to be the method showing the best history of success among the three remaining methods. Then within each method the same kind of selection process with respect to orders is used. In this way the past performance of the different methods and orders influences the choice of which are allowed to compete, such that the more successful have a higher probability of being selected.

B. The Selection Process

There are four methods available for the integration process, and within each method there are four orders available. The methods and orders are as follows:

1) The Adams method with orders 4(4), 5(4), 6(4), 7(5).
2) The Butcher formulas with orders 3(4), 5(4), 7(4), 7(5).
3) The Cowell method with orders 7(5), 9(5), 11(5), 13(5).
4) The Shanks formulas with orders 4, 5, 6, 7.

Each order of each multistep method has an associated Runge-Kutta-Shanks restart procedure order given in parenthesis after the method order. Details on these methods are given in Chapter II of this report. The magnetic tape containing the coefficients has several additional orders of each method, but the program is now set to use just those mentioned above.

The selection process is the following. The first of two methods is chosen at random (using a random number generator) from among the four available. The second method is chosen to be the method showing the best history of success among the three remaining methods, with the cumulative history file being used to determine the degree of success. Then within each method the same kind of selection process with respect to orders is used. That is, the first order is chosen at random, and the second order is chosen on the basis of which of the remaining three has been the most successful (fastest running) order of that method. Thus it is seen that the past performance of the different methods and orders influences the choice of which are allowed to compete, such that the more successful have a higher probability of being selected.

In using the time as the sole estimate of performance efficiency, it is assumed that all orders and methods have satisfactorily met the accuracy requirements. The accuracy requirements of each method are met by controlling step size and making error estimates at each step. The method of error estimate is different for the different methods. In the Runge-Kutta-Shanks single step method, the error is estimated by taking two half steps and then a whole step. In the Adams and Butcher methods the difference between predictor and corrector is used. In the Cowell method a mid-range formula is used. (Only in the Adams and the Runge-Kutta-Shanks cases is there good theoretical justification for using these methods to calculate the actual error -- the error estimates in the Butcher and Cowell methods are essentially empirical.)

C. Organization of the History File

The history of the effectiveness of each method is recorded in a file called "A831HST" and organized in the following manner.

Associated with each order of each method are two numbers. The first (a positive number) records the time associated with trials in which this order was the winner. The second (a negative number) records the time associated with trails in which it was the loser. The sum of these two numbers is taken as the "score" or performance number and will be greater if the order of this method has been a consistent winner and will be less (more negative) if it has been a consistent loser.

Associated with each method then is a method score analogous to the order scores just described. That is, each method has one positive and one negative number recording the time spent winning and losing respectively. In addition to this, a history is also kept of which methods the wins and losses were against, but this part of the history is not used in selecting competitors.

The history file is printed in an output file called "HISTORY." In describing this, use will be made of an abbreviated notation. A stands for Adams method, B for Butcher, C for Cowell, and S for Shanks formulas. A number given following the letter designates the order of that method where 1 stands for the lowest order available, 2 for the next lowest, etc. Thus A3 stands for the second highest order Adams method. A sign following the letter or number designates winning time or losing time for this method-order. For example, $B2^+$ designates winning time for Butcher's method, second lowest order; C^- designates losing time for Cowell's method; etc. Finally, if a letter follows the sign in parenthesis, this designates which method the win or loss was against; thus $B^+(A)$ designates winning time by Butcher against Adams. With this notation the organization of the history file is as follows:

The first three items (printed on the first line of the output of the history file) are not times but other bookkeeping items. The first number gives the date (in the form year, day) that this particular history file was initialized, that is the date the tape was first created. The second number gives the total number of times the procedure has been called (using this particular history file). The third number gives the present value of the random number used in generating the random number sequence.

Following these three numbers come the cumulative times the various methods spent winning and losing. These are organized in a 9 row - 8 column matrix. The first 4 rows give wins and losses of the various orders within each method, that is the results of the competitive trials over the 1/8 sections of the range of integration. Table I gives this organization in terms of the notation described above.

Following this is a row giving cumulative winning and losing times by methods; that is, the results of the trials over the 1/4 sections of the range of integration. This row is organized:

$$A^+ \qquad A^- \qquad B^+ \qquad B^- \qquad C^+ \qquad C^- \qquad S^+ \qquad S^-.$$

The last four rows give a more detailed breakdown of the line above, giving the method against which the winning and losing times were made. It is organized as in Table II. It is noted here that entries of the form $A^+(A)$, $B^-(B)$, C^+C, etc. will all be zero, since methods do not compete against themselves.

TABLE I

ORGANIZATION OF CUMULATIVE WINNING AND LOSING TIMES BY METHOD AND ORDER

$A1^{+}$	$A1^{-}$	$A2^{+}$	$A2^{-}$	$A3^{+}$	$A3^{-}$	$A4^{+}$	$A4^{-}$
$B1^{+}$	$B1^{-}$	$B2^{+}$	$B2^{-}$	$B3^{+}$	$B3^{-}$	$B4^{+}$	$B4^{-}$
$C1^{+}$	$C1^{-}$	$C2^{+}$	$C2^{-}$	$C3^{+}$	$C3^{-}$	$C4^{+}$	$C4^{-}$
$S1^{+}$	$S1^{-}$	$S2^{+}$	$S2^{-}$	$S3^{+}$	$S3^{-}$	$S4^{+}$	$S4^{-}$

Notation here: A = Adams, B = Butcher, C = Cowell, S = Shanks;

1 = lowest order, 2 = second lowest order, etc;

+ stands for win, - stands for loss.

TABLE II

ORGANIZATION OF CUMULATIVE WINNING AND LOSING TIMES BY METHOD VS. METHOD

$A^{+}(A)$	$A^{-}(A)$	$B^{+}(A)$	$B^{-}(A)$	$C^{+}(A)$	$C^{-}(A)$	$S^{+}(A)$	$S^{-}(A)$
$A^{+}(B)$	$A^{-}(B)$	$B^{+}(B)$	$B^{-}(B)$	$C^{+}(B)$	$C^{-}(B)$	$S^{+}(B)$	$S^{-}(B)$
$A^{+}(C)$	$A^{-}(C)$	$B^{+}(C)$	$B^{-}(C)$	$C^{+}(C)$	$C^{-}(C)$	$S^{+}(C)$	$S^{-}(C)$
$A^{+}(S)$	$A^{-}(S)$	$B^{+}(S)$	$B^{-}(S)$	$C^{+}(S)$	$C^{-}(S)$	$S^{+}(S)$	$S^{-}(S)$

Notation here: A = Adams, B = Butcher, C = Cowell, S = Shanks;

+ stands for win, - stands for loss.

$A^{+}(S)$ stands for Adams win against Shanks,

$C^{-}(B)$ stands for Cowell loss against Butcher, etc.

Entries of the form $A^{+}(A)$, or $C^{-}(C)$ etc., should all be zero since a method does not compete against itself.

D. Inputs to the Executive Procedure

A call in the executive procedure would look like the following:

DIFEQINT (N, XI, XF, Y, F, P, EA, ER, DX)

Here the identifiers in parenthesis are the inputs to the procedure and represent the following information:

N is the number of equations in the system to be integrated,

XI is the initial value of the independent variable,

XF is the final value of the independent variable,

Y is the initial values of the dependent variables. Y is a vector (one dimensional array). At the conclusion of the procedure Y is set to the final values of the dependent variable; that is, Y is also the output variable.

F is the procedure for calculating dy/dx as a function of x and y. This procedure must be written by the user and describes the system of differential equations being integrated. It must be written so as to have four parameters:

(a) N, the number of equations,

(b) X, the independent variable,

(c) Y, the dependent variable (vector),

(d) FV, the vector values of dy/dx at the point x, y.

The first three parameters are input and FV is the output.

P is the error accumulation parameter. This parameter expresses the user's opinion as to how the errors are going to accumulate over the range of integration. For example, if it is expected that the errors will be random then P would be set to 0.5. If it is expected that the errors will accumulate linearly then set P = 1. These are the two most usual cases but other situations can occur.

EA is the absolute error vector. This vector gives the acceptable absolute errors in the value of Y final.

ER is the relative error vector. This vector gives the acceptable relative error in the value of Y final. It is the weaker of the two conditions EA and ER that is met for each component of the vector Y.

DX is the estimated value of the initial step size. This estimate need not be especially accurate since the individual methods will adjust the step size to the appropriate value.

E. Updating of the History File and Forgetting

The times recorded in the history file are cumulative. That is, after a competition is held, the times taken by the competing methods and orders are added to (for a win) or subtracted from (for a loss) the appropriate positions in the history file. Thus, the entries in the history tables represent an index expressing the cumulative past performance.

Decisions as to which method or order within a method is considered to have the best performance history are made on the basis of the sum of the win and loss entries for that method or order. The method or order having the maximum value for this sum is considered to have best history (remembering that the loss entries are negative). One notes that not all the history file is used in the decision making process; in particular, those entries in Table II are not used in decision making but are recorded only to give the user a more detailed account of the competitions.

One further feature is introduced into the learning process and this is the gradual "forgetting" of events in the more distant past. This is accomplished by multiplying those history scores used in the decision making by a factor less than one, just before the most recent histories are added.

This causes the events in the distant past to have less influence than those more recent in determining the performance figure of an order and method. The factor used is 0.98 but it is not known what would be the optimum factor. Note that the entries in the history file described in Table II do not involve forgetting. Since these entries are not used in any decision making process but only tabulated for the user's interest, forgetting would serve no practical purpose here. The entries in Table II represent then a total or unattenuated history of the competition between the various methods.

F. Reading of Coefficients and History Files

Also needed as input to the executive routine are the tables of coefficients associated with the various methods and the past performance history file. These are read in to the procedure the first time the procedure is called and a flag set (in an element of an array declared OWN) to indicate that these have been read in once. This information is stored in an array declared OWN and need not be read in again during the operation of the program.

The coefficients are stored on a tape file called "TAPE831." It contains the following coefficients:

Adams'method, orders 4 through 10,

Butcher's formulas, orders 3, 5, 7, 9, 11,

Cowell's method, orders 7, 9, 11, 13,

Shanks formulas, orders 4, 5, 6, 7, 7, 8, 8.

Only four orders of each method are actually used.

The history is stored in a tape file called "A831HST." This tape must be mounted with a write ring and is updated every time the procedure is called.

G. Outputs of the Executive Procedure

The executive procedure returns the final value of the independent variable as its principal output. This is returned through the same variable, Y (a vector), described in Inputs to the Procedure, paragraph D of this Chapter.

There are several other types of output that are printed. First, when the procedure is called for the first time and reads in the past performance history file, it prints out this history in a print file called "HISTORY".

Also printed out in this file is a pair of numbers giving the method and order that is about to be used and the times for each order and method after the comparisons have been made. This information is printed in a two digit code, the first digit representing the method and the second (if present) indicating the order. The method code is:

0 represents Adams,

1 represents Butcher,

2 represents Cowell,

3 represents Shanks.

The order code is such that 0 represents the lowest order, 1 represents the next lowest order, etc.

Also printed in the file "HISTORY" are the results of comparison runs in which the results (values of the dependent variable) of the two competing method orders differ by more than twice the allowed errors. Also printed are the initial and final values of the independent variable, the two differing values of the dependent variables and an integer telling which component of the dependent variable appears to be in error.

Other messages associated with anomolous conditions are also printed in this file. In particular an integer overflow condition occurs if the step size collapses. Recovery from step size collapse can usually be effected but the message "INTEGER OVERFLOW" will be printed in file "HISTORY" whenever it occurs.

Finally, the procedure outputs the updated performance history by writing it back into the "A831HST" tape file.

H. Flow Diagram and Program Listing

Figure 6 is the flow diagram for the executive procedure. The program listing for the executive procedure follows at the end of this section. Since the individual methods and restart programs are also listed elsewhere in this report, their listing here is given in "squeezed" form.

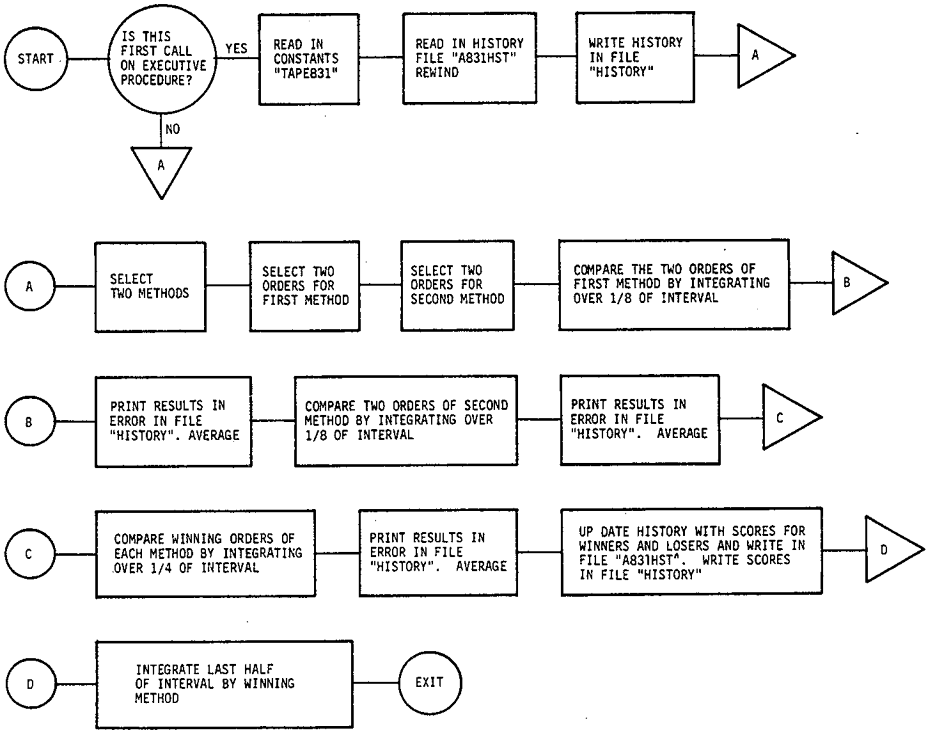
START
IS THIS FIRST CALL ON EXECUTIVE PROCEDURE?
YES
NO
READ IN CONSTANTS "TAPE831"
READ IN HISTORY FILE "A831HST" REWIND
WRITE HISTORY IN FILE "HISTORY"
A
A
A
SELECT TWO METHODS
SELECT TWO ORDERS FOR FIRST METHOD
SELECT TWO ORDERS FOR SECOND METHOD
COMPARE THE TWO ORDERS OF FIRST METHOD BY INTEGRATING OVER 1/8 OF INTERVAL
B
B
PRINT RESULTS IN ERROR IN FILE "HISTORY". AVERAGE
COMPARE TWO ORDERS OF SECOND METHOD BY INTEGRATING OVER 1/8 OF INTERVAL
PRINT RESULTS IN ERROR IN FILE "HISTORY". AVERAGE
C
C
COMPARE WINNING ORDERS OF EACH METHOD BY INTEGRATING OVER 1/4 OF INTERVAL
PRINT RESULTS IN ERROR IN FILE "HISTORY". AVERAGE
UP DATE HISTORY WITH SCORES FOR WINNERS AND LOSERS AND WRITE IN FILE "A831HST". WRITE SCORES IN FILE "HISTORY"
D
D
INTEGRATE LAST HALF OF INTERVAL BY WINNING METHOD
EXIT

Figure 6. Flow Diagram for the Executive Procedure.

```
  FILE PUN 15 "HISTORY"(3,15);                                             00002000
  FILE PUNCH 0(1,10),A831HST 2(2,75,SAVE 90);                              00003000
  PROCEDURE DIFEQINT(N,XI,XF,Y,F,P,EA,ER,DX);                              00004000
  VALUE N,XF,XI,DX,P;                                                      00005000
  INTEGER N;                                                               00006000
  REAL XI,XF,DX,P;                                                         00007000
  PROCEDURE F;                                                             00008000
  ARRAY Y,EA,ER[0];                                                        00009000
  IF XI ≠XF THEN                                                           00010000
  BEGIN                                                                    00011000
INTEGER PROCEDURE START(N,XI,XF,C1,EA,ER,F,M,X,YIV,YH,FH,YFV,CYI,CYM,PA,0000
P,FNEVAL,RKSCONST);VALUE N,XI,XF,C1,M,CYI,CYM,PA,P,FNEVAL ;INTEGER N,C1,0000
M,CYI,CYM,PA,FNEVAL ;REAL XI,XF,X,P ;REAL ARRAY EA,ER,YIV,YFV,RKSCONST[00000
],YH,FH[0,0];PROCEDURE F ;BEGIN INTEGER I,J,K,L,COEFFCNT,FNMAX,INDX,NIND0000
X,CNTR ;REAL INT,H,TWOH,T1 ;REAL ARRAY HC,TWOHC[0:(((FNEVAL+3)×FNEVAL)/20000
)-2],EAV,ERV,Y1,Y2,Y3,Y4[0:N],G[0:FNEVAL -2,0:N];LABEL L1,L2,L3,L4,L5,L60000
 ;PROCEDURE RUNKUT(N,X,FNMAX,COEFF,YIV,YFV,FV,F,G);VALUE N,X,FNMAX ;INTE0000
GER N,FNMAX ;REAL X ;REAL ARRAY COEFF,YIV,YFV,FV[0],G[0,0];PROCEDURE F ;0000
BEGIN INTEGER I,J,K,CCNT ;REAL TEMP ;TEMP :=COEFF[CCNT :=0];FOR I :=0 ST0000
EP 1 UNTIL FNMAX DO BEGIN FOR J :=1 STEP 1 UNTIL N DO YFV[J]:=FV[J]×TEMP0000
 +YIV[J];FOR K :=0 STEP 1 UNTIL I -1 DO BEGIN TEMP :=COEFF[CCNT :=CCNT +0000
1];FOR J :=1 STEP 1 UNTIL N DO YFV[J]:=G[K,J]×TEMP +YFV[J]END ;F(N,X +CO0000
EFF[CCNT :=CCNT +1],YFV,G[I,*]);TEMP :=COEFF[CCNT :=CCNT +1]END ;FOR J :0000
=1 STEP 1 UNTIL N DO YFV[J]:=FV[J]×TEMP +YIV[J];FOR K :=0 STEP 1 UNTIL F0000
NMAX DO BEGIN TEMP :=COEFF[CCNT :=CCNT +1];FOR J :=1 STEP 1 UNTIL N DO Y0000
FV[J]:=G[K,J]×TEMP +YFV[J]END ;END ;BOOLEAN PROCEDURE COMP(N,EAV,ERV,Y,Z0000
);VALUE N ;INTEGER N ;REAL ARRAY EAV,ERV,Y,Z[0];BEGIN INTEGER J ;REAL T10000
 ;LABEL L1 ;FOR J :=1 STEP 1 UNTIL N DO IF (T1 :=ABS(Y[J]-Z[J]))>EAV[J]T0000
HEN BEGIN IF T1 >ERV[J]×ABS(Z[J])THEN BEGIN COMP :=FALSE ;GO TO L1 END E0000
ND ;COMP :=TRUE ;L1:END ;CNTR :=1 ;IF M ≠0 THEN BEGIN COEFFCNT :=(((FNEV0000
AL +3)×FNEVAL)/2)-2 ;FNMAX :=FNEVAL -2 ;INT :=XF -XI ;TWOH :=(INT +INT)/0000
C1 ;H :=INT /C1 ;FOR I :=0 STEP 1 UNTIL COEFFCNT DO BEGIN HC[I]:=(T1 :=R0000
KSCONST[I])×H ;TWOHC[I]:=T1 ×TWOH END ;INDX :=CYI MOD CYM ;T1 :=(C1 /2)*0000
P ;FOR J :=1 STEP 1 UNTIL N DO BEGIN EAV[J]:=EA[J]/T1 ;ERV[J]:=ER[J]/T1 0000
END ;IF PA =2 THEN BEGIN NINDX :=(INDX +1)MOD CYM ;K :=0 ;GO TO L3 END ;0000
IF PA =1 THEN L :=M DIV 2 ;RUNKUT(N,X,FNMAX,TWOHC,YIV,Y1,FH[INDX,*],F,G)0000
;GO TO L4 ;L1:TWOH :=H ;CNTR :=CNTR +CNTR ;C1 :=C1 +C1 ;H :=INT /C1 ;FOR0000
 I :=0 STEP 1 UNTIL COEFFCNT DO BEGIN TWOHC[I]:=HC[I];HC[I]:=RKSCONST[I]0000
×H END ;INDX :=CYI MOD CYM ;T1 :=(C1 /2)*P ;IF PA =2 THEN BEGIN K :=0 ;N0000
INDX :=(INDX +1)MOD CYM ;FOR J :=1 STEP 1 UNTIL N DO BEGIN EAV[J]:=EA[J]0000
```

```
/T1 ;ERV[J]:=ER[J]/T1 ;Y1[J]:=YH[NINDX,J]END ;L2:RUNKUT(N,K ×H +X,FNMAX,0000
HC,YH[INDX,*],YH[NINDX,*],FH[INDX,*],F,G);K :=K +1 ;INDX :=NINDX ;NINDX 0000
:=(INDX +1)MOD CYM ;F(N,K ×H +X,YH[INDX,*],FH[INDX,*]);RUNKUT(N,K ×H +X,0000
FNMAX,HC,YH[INDX,*],YH[NINDX,*],FH[INDX,*],F,G);IF COMP(N,EAV,ERV,Y1,YH[0000
NINDX,*])THEN BEGIN K :=K +1 ;IF K <M THEN BEGIN INDX :=NINDX ;NINDX :=(0000
INDX +1)MOD CYM ;F(N,K ×H +X,YH[INDX,*],FH[INDX,*]);L3:RUNKUT(N,K ×H +X,0000
FNMAX,TWOHC,YH[INDX,*],Y1,FH[INDX,*],F,G);GO TO L2 END ;IF K =M THEN BEG0000
IN INDX :=NINDX ;F(N,K ×H +X,YH[INDX,*],FH[INDX,*])END END ELSE GO TO L10000
 END ELSE BEGIN FOR J :=1 STEP 1 UNTIL N DO BEGIN EAV[J]:=EA[J]/T1 ;ERV[0000
J]:=ER[J]/T1 ;Y1[J]:=Y2[J]END ;L4:RUNKUT(N,X,FNMAX,HC,YIV,Y2,FH[INDX,*],0000
F,G);INDX :=(INDX +1)MOD CYM ;F(N,X +H,Y2,FH[INDX,*]);RUNKUT(N,X +H,FNMA0000
X,HC,Y2,Y4,FH[INDX,*],F,G);K :=2 ;IF PA =0 THEN BEGIN L5:IF COMP(N,EAV,E0000
RV,Y1,Y4)THEN BEGIN IF K <M THEN BEGIN INDX :=(INDX +1)MOD CYM ;F(N,K ×H0000
 +X,Y4,FH[INDX,*]);RUNKUT(N,K ×H +X,FNMAX,TWOHC,Y4,Y1,FH[INDX,*],F,G);RU0000
NKUT(N,K ×H +X,FNMAX,HC,Y4,Y3,FH[INDX,*],F,G);K :=K +1 ;INDX :=(INDX +1)0000
MOD CYM ;F(N,K ×H +X,Y3,FH[INDX,*]);RUNKUT(N,K ×H +X,FNMAX,HC,Y3,Y4,FH[I0000
NDX,*],F,G);K :=K +1 ;GO TO L5 END ;IF K =M THEN BEGIN INDX :=(INDX +1)M0000
OD CYM ;F(N,K ×H +X,Y4,FH[INDX,*]);FOR J :=1 STEP 1 UNTIL N DO YFV[J]:=Y0000
4[J]END ELSE FOR J :=1 STEP 1 UNTIL N DO YFV[J]:=Y3[J]END ELSE GO TO L1 0000
END ELSE BEGIN L6:IF COMP(N,EAV,ERV,Y1,Y4)THEN BEGIN INDX :=(INDX +1)MOD0000
 CYM ;F(N,K ×H +X,Y4,FH[INDX,*]);IF K <M THEN BEGIN IF K =L THEN FOR J :0000
=1 STEP 1 UNTIL N DO YFV[J]:=Y4[J];RUNKUT(N,K ×H +X,FNMAX,TWOHC,Y4,Y1,FH0000
[INDX,*],F,G);RUNKUT(N,K ×H +X,FNMAX,HC,Y4,Y3,FH[INDX,*],F,G);K :=K +1 ;0000
INDX :=(INDX +1)MOD CYM ;F(N,K ×H +X,Y3,FH[INDX,*]);IF K =L THEN FOR J :0000
=1 STEP 1 UNTIL N DO YFV[J]:=Y3[J];RUNKUT(N,K ×H +X,FNMAX,HC,Y3,Y4,FH[IN0000
DX,*],F,G);K :=K +1 ;GO TO L6 END ;FOR J :=1 STEP 1 UNTIL N DO YIV[J]:=Y0000
4[J]END ELSE GO TO L1 END END ;X :=M ×H +X END ;START :=CNTR ;END ;PROCE0000
DURE ADAMS(N,XI,XF,Y,F,P,Q,DX,EA,ER,ADAMSCOEFF,RKSFNS,RKSORDER,RKSCOEFF,0000
START,SHANKS);VALUE N,XI,XF,P,Q,DX,RKSFNS,RKSORDER;REAL XI,XF,P,DX;INTEG0000
ER N,Q,RKSFNS,RKSORDER;REAL ARRAY Y,EA,ER,ADAMSCOEFF,RKSCOEFF[0];PROCEDU0000
RE F,SHANKS;INTEGER PROCEDURE START;BEGIN DEFINE YI=Y#,YB=Y#,YF=Y#,POINT0000
S=BEGIN IF NOT BGOOD THEN ALLI YB[I]←YD[I];X←XF-INTERVAL×(C2/C1);MULT←ST0000
ART(N,XI,XF,C1,EA,ER,F,QMINUS1,X,YB,FH,FH,YB,J,QT2M1,0,P,RKSFNS,RKSCOEFF0000
);ALLI C[I]←YP[I]←YB[I];BGOOD←TRUE;END#,CALLOB=BEGIN SHANKS(N,X,XF,YB,F,0000
RKSFNS,RKSORDER,RKSCOEFF,P,EA,ER,XF-X);END#,ALLMU=FOR MU←0STEP 1UNTIL QM0000
INUS1 DO#,ALLI=FOR I←1STEP 1UNTIL N DO#,RESET=BEGIN DC←0;PC←Q;CU←(C1*(-P0000
))×GR;H←INTERVAL/C1;ALLMU BEGIN HB[MU]←B[MU]×H;HBS[MU]←BS[MU]×H;END;HBSQ0000
Z←BSQZ×H;ALLI BEGIN EAL[I]←(EAU[I]←EA[I]×CU)×C2MQP5;ERL[I]←(ERU[I]←ER[I]0000
×CU)×C2MQP5;HAL[I]←ABS(EAL[I]/HBSQZ);HRL[I]←ABS(ERL[I]/HBSQZ);END;END#;L0000
ABEL RESTART,NEXTSTEP,FLIP,FLOP,TEST,DUBBLE,HALF,FINISH;REAL ARRAY FH[0:0000
(2×Q-2),0:N],B,BS,HB,HBS[0:Q-1],C,YP,YC,YD,FP,FC,EAU,EAL,ERU,ERL,HAL,HRL0000
[0:N];REAL H,X,CU,C2,GR,YCI,BSQZ,FHJI,FMUI,HBMU,M1DP,HBSMU,HBSQZ,ERROR,C0000
HANGE,C2MQP5,INTERVAL;INTEGER I,J,K,C1,DC,PC,MU,MULT,JZERO,QT2M1,QMINUS10000
,QTIMES2;BOOLEAN BGOOD,FLIPPED,TOOSMALL;C2MQP5←1/2*(Q+5);QMINUS1←Q-1;QTI0000
MES2←Q+Q;QT2M1←QTIMES2-1;ALLMU BEGIN B[MU]←ADAMSCOEFF[MU];BS[MU]←ADAMSCO0000
EFF[MU+Q];END;BSQZ←ADAMSCOEFF[QTIMES2];GR←ADAMSCOEFF[QTIMES2+1];C1←1;H←I0000
NTERVAL←XF-XI;DX←ABS(DX);WHILE ABS(H)>DX OR C1<Q DO BEGIN C1←C1+C1;H←INT0000
ERVAL/C1;END;C2←C1;JZERO←J←0;F(N,XI,YI,FH[0,*]);X←XI;ALLI YB[I]←YI[I];BG0000
OOD←TRUE;RESTART:POINTS;C1←C1×MULT;C2←C2×MULT-Q;IF(J←JZERO-1)<0THEN J←J+0000
QT2M1;RESET;NEXTSTEP:X←XF-C2×H;ALLMU BEGIN IF(J←J+1)=QT2M1 THEN J←0;HBMU0000
←HB[MU];HBSMU←HBS[MU];ALLI BEGIN YP[I]←(FMUI←FH[J,I])×HBMU+YP[I];C[I]←HB0000
SMU×FMUI+C[I];END;END;F(N,X,YP,FP);FLIP:ALLI YC[I]←FP[I]×HBSQZ+C[I];F(N,0000
X,YC,FC);ALLI IF(CHANGE←ABS(FC[I]-FP[I]))>HAL[I]THEN IF CHANGE>ABS(HRL[I0000
]×FC[I])THEN GO TO FLOP;FLIPPED←TRUE;GO TO TEST;FLOP:ALLI YC[I]←FC[I]×HB0000
SQZ+C[I];F(N,X,YC,FP);ALLI IF(CHANGE←ABS(FP[I]-FC[I]))>HAL[I]THEN IF CHA0000
NGE>ABS(HRL[I]×FP[I])THEN GO TO FLIP;FLIPPED←FALSE;TEST:IF(J←J+1)=QT2M1 0000
THEN J←0;TOOSMALL←TRUE;ALLI BEGIN FHJI←FH[J,I]←IF FLIPPED THEN FC[I]ELSE0000
 FP[I];ERROR←ABS(YP[I]-(YCI←C[I]←YP[I]←(FHJI×HBSQZ+C[I])));IF BGOOD THEN0000
 YD[I]←YCI ELSE YB[I]←YCI;YCI←ABS(YCI);IF ERROR>EAU[I]THEN IF ERROR>ERU[0000
I]×YCI THEN GO TO HALF;IF ERROR>EAL[I]THEN IF ERROR>ERL[I]×YCI THEN TOOS0000
MALL←FALSE;END;PC←PC+1;IF BGOOD THEN BGOOD←FALSE ELSE BGOOD←TRUE;IF C2≥10000
.0THEN C2←C2-1.0ELSE GO TO FINISH;IF TOOSMALL THEN BEGIN DC←DC+1;IF DC≥30000
THEN IF PC≥QT2M1 THEN IF C2≥1THEN GO TO DUBBLE;IF(JZERO←(J←JZERO)+1)=QT20000
M1 THEN JZERO←0;GO TO NEXTSTEP;END;DC←0;IF(JZERO←(J←JZERO)+1)=QT2M1 THEN0000
 JZERO←0;GO TO NEXTSTEP;DUBBLE:C1←C1 DIV 2;C2←(C2-1.0)/2.0;RESET;K←J;FOR0000
 MU←1STEP 1UNTIL QMINUS1 DO BEGIN IF(J←J-1)<0THEN J←J+QT2M1;IF(K←K-2)<0T0000
HEN K←K+QT2M1;ALLI FH[J,I]←FH[K,I];END;IF(J←(JZERO←J)-1)<0THEN J←J+QT2M10000
;GO TO NEXTSTEP;HALF:IF(J←J-1)<0THEN J←J+QT2M1;JZERO←J;C1←C1+C1;IF(C2←C20000
+C2+2.0)<Q THEN GO TO FINISH;GO TO RESTART;FINISH:IF NOT FLIPPED THEN AL0000
LI FC[I]←FP[I];IF NOT BGOOD THEN ALLI YB[I]←YD[I];X←XF-INTERVAL×(C2/C1);0000
IF C2≠0THEN CALLOB;ALLI YF[I]←YB[I];END;PROCEDURE BUTCHER(N,XI,XF,K,EA,E0000
R,DX,CON,FUNCTION,EX,RKC,START,SHANKS,YIV,RKSNF,RKSODR);VALUE N,XI,XF,K,0000
DX,CON,EX,RKSNF,RKSODR;REAL ARRAY YIV[0];INTEGER RKSNF,RKSODR;INTEGER N,0000
K;PROCEDURE FUNCTION,SHANKS;INTEGER PROCEDURE START;REAL XI,XF,DX,EX;REA0000
L ARRAY RKC[0];REAL ARRAY CON,EA,ER[0];BEGIN REAL ARRAY Y,F[0:16,0:N];RE0000
AL SC1,X;REAL DX2;REAL DX1,COA,COB,COLA,COLB,COGA,COGB,TEST,TEMPY,TEMPF,0000
A1,A2,A3,C2;INTEGER I,J,CYL,INDEX,C1,M;INTEGER CYL3;REAL ARRAY SUMYIP,SU0000
MYP,SUMYC,FV1[0:N];LABEL STRRT,RESTART,FINISH;REAL P2,T1,T2;INTEGER COUN0000
T,TOTCNT,CYL1,CYL2,M1;LABEL DUBSRT;REAL ARRAY COO[0:3×K];INTEGER CYO;INT0000
EGER COUNTER;INTEGER KM1;REAL OMT,K6,K61,K62;REAL INTV;INTEGER KM3,J2,J30000
```

```
,J6;REAL XDXT,XDX;REAL ARRAY RE,AE[0:N];FOR I←1STEP 1UNTIL N DO Y[0,I]←Y
IV[I];IF K=1OR K=2OR K=3THEN OMT←0.5ELSE OMT←2/3;K6←6×K;K61←(6×K)+1;K62←
(6×K)+2;KM1←K-1;INTV←XF-XI;X←XI;C1←1;WHILE(C1<K+1)OR((ABS(INTV)/C1)>ABS(
DX))DO C1←C1+C1;C2←C1;P2←1/(2*((2×K)+4));CYL←0;CYO←0;TOTCNT←0;FUNCTION(N
,XI,Y[0,*],F[0,*]);RESTART:COUNTER←KM1;C1←C1×(I←START(N,XI,XF,C1,EA,ER,F
UNCTION,KM1,X,YIV,Y,F,YIV,CYO,16,2,EX,RKSNF,RKC));C2←C2×I-KM1;CYL←CYO;DU
BSRT:DX←INTV/C1;KM3←3×KM1;FOR J←0STEP 3UNTIL KM3 DO BEGIN J2←2×J;COO[J]←
CON[J2+1]×DX;COO[J+1]←CON[J2+3]×DX;COO[J+2]←CON[J2+5]×DX;END;SC1←C1*EX;F
OR I←1STEP 1UNTIL N DO BEGIN AE[I]←EA[I]/SC1;RE[I]←ER[I]/SC1;END;A1←CON[
K6]×DX;A2←CON[K61]×DX;A3←CON[K62]×DX;STRRT:XDXT←X+(DX×OMT);XDX←X+DX;FOR 
I←1STEP 1UNTIL N DO SUMYIP[I]←SUMYP[I]←SUMYC[I]←0;FOR J←0STEP 1UNTIL KM1
 DO BEGIN CYL3←(KM1-J+CYL)MOD 16;J3←3×J;J6←6×J;COA←CON[J6];COB←COO[J3];C
OLA←CON[J6+2];COLB←COO[J3+1];COGA←CON[J6+4];COGB←COO[J3+2];FOR I←1STEP 1
UNTIL N DO BEGIN TEMPY←Y[CYL3,I];TEMPF←F[CYL3,I];SUMYIP[I]←SUMYIP[I]+(CO
A×TEMPY)+(COB×TEMPF);SUMYP[I]←SUMYP[I]+(COLA×TEMPY)+(COLB×TEMPF);SUMYC[I
]←SUMYC[I]+(COGA×TEMPY)+(COGB×TEMPF);END;END;FUNCTION(N,XDXT,SUMYIP,FV1)
;FOR I←1STEP 1UNTIL N DO BEGIN TEMPF←FV1[I];SUMYP[I]←SUMYP[I]+(A1×TEMPF)
;SUMYC[I]←SUMYC[I]+(A2×TEMPF);END;FUNCTION(N,XDX,SUMYP,FV1);CYL←(CYL+1)M
OD 16;CYO←(CYL+KM1)MOD 16;COUNT←0;FOR I←1STEP 1UNTIL N DO BEGIN TEMPY←SU
MYC[I]+(A3×FV1[I]);T1←AE[I];T2←ABS(RE[I]×TEMPY);TEST←ABS(TEMPY-SUMYP[I])
;IF TEST>T1 AND TEST>T2 THEN BEGIN C2←C2+C2;CYL←(CYL+15)MOD 16;CYO←(CYL+
KM1)MOD 16;IF C2<KM1 THEN BEGIN SHANKS(N,X,XF,Y[CYO,*],FUNCTION,RKSNF,RK
SODR,RKC,EX,EA,ER,DX);GO TO FINISH;END;C1←C1+C1;GO TO RESTART;END;Y[CYO,
I]←TEMPY;IF TEST<P2×T1 OR TEST<P2×T2 THEN COUNT←COUNT+1;END;C2←C2-1;X←XF
-(DX×C2);IF C2=0THEN GO TO FINISH;IF C2<1THEN BEGIN SHANKS(N,X,XF,Y[CYO,
*],FUNCTION,RKSNF,RKSODR,RKC,EX,EA,ER,DX);GO TO FINISH;END;FUNCTION(N,X,
Y[CYO,*],F[CYO,*]);IF COUNT=N THEN BEGIN TOTCNT←TOTCNT+1;IF TOTCNT≥3THEN
 BEGIN IF COUNTER≥2×K THEN BEGIN COUNTER←0;C2←C2/2;C1←C1/2;IF C2<1THEN B
EGIN SHANKS(N,X,XF,Y[CYO,*],FUNCTION,RKSNF,RKSODR,RKC,EX,EA,ER,DX);GO TO
 FINISH;END;TOTCNT←0;FOR I←1STEP 1UNTIL N DO FOR J←1STEP 1UNTIL KM1 DO B
EGIN CYL1←(CYO+16-J)MOD 16;CYL2←(CYO+16-(2×J))MOD 16;Y[CYL1,I]←Y[CYL2,I]
;F[CYL1,I]←F[CYL2,I];END;GO TO DUBSRT;END;END;END;COUNTER←COUNTER+1;GO T
O STRRT;FINISH:FOR I←1STEP 1 UNTIL N DO YIV[I]←Y[CYO,I];END BUTCHER;PROC
EDURE COWELL(N,XI,XF,Y,F,EA,ER,P,DX,RKSFN,RKSORDER,RKSCOEFF,Q,COWELLCOEF
F,START,SHANKS);VALUE N,XI,XF,P,DX,RKSFN,RKSORDER,Q ;INTEGER N,RKSFN,RKS
ORDER,Q ;REAL XI,XF,P,DX ;REAL ARRAY Y,EA,ER,RKSCOEFF,COWELLCOEFF[0];PRO
CEDURE F,SHANKS ;INTEGER PROCEDURE START ;BEGIN INTEGER C1,M,MM1,QP1,TQP
1,INDX,I1,I2,I3,I,J,K,CYI ;INTEGER CORRECTCNT ;REAL INT,C2,DFACTOR,X,H,T
1,T2,T3,T4,T5,T6 ;BOOLEAN DFLAG,PFLAG ;REAL ARRAY FH[0:Q +Q,0:N],YMID1,Y
P,YC,YM,CS,FP,HDM1F,HDM1FMID,EAV,ERV,EAVD,ERVD[0:N],PCOEFF,CCOEFF,MCOEFF
```

```
[0:Q];LABEL L0,L1,L2,L3,L4,L5,STARTER,DOUBLER,ACCEPT,CORRECT,CLOSER;INT 0000 154
:=XF -XI ;C1 :=1 ;L0:C1 :=C1 +C1 ;IF C1 ≤Q THEN GO TO L0 ;IF DX <0 THEN 0000 155
DX :=-DX ;IF DX ≠0 THEN BEGIN L1:IF ABS(INT)/C1 >DX THEN BEGIN C1 :=C1 +0000 156
C1 ;GO TO L1 END END ;C2 :=C1 ;M :=Q DIV 2 ;MM1 :=M -1 ;QP1 :=Q +1 ;TQP10000 157
 :=QP1 +Q ;DFACTOR :=2.0 *(Q +3);INDX :=0 ;X :=XI ;F(N,X,Y,FH[0,*]);STAR0000 158
TER:C1 :=(I1 :=START(N,XI,XF,C1,EA,ER,F,Q,X,Y,FH,FH,YMID1,INDX,TQP1,1,P,0000 159
RKSFN,RKSCOEFF))×C1 ;C2 :=C2 ×I1 -Q ;INDX :=(INDX +Q)MOD TQP1 ;IF C2 <M 0000 160
THEN GO TO CLOSER ;DOUBLER:H :=INT /C1 ;FOR K :=0 STEP 1 UNTIL Q DO BEGI0000 161
N PCOEFF[K]:=COWELLCOEFF[K]×H ;CCOEFF[K]:=COWELLCOEFF[I1 :=K +QP1]×H ;MC0000 162
OEFF[K]:=COWELLCOEFF[I1 +QP1]×H END ;T1 :=(C1 *P)×10.0 ;FOR J :=1 STEP 10000 163
 UNTIL N DO BEGIN EAVD[J]:=(EAV[J]:=EA[J]/T1)/DFACTOR ;ERVD[J]:=(ERV[J]:0000 164
=ER[J]/T1)/DFACTOR END ;T1 :=MCOEFF[0];FOR J :=1 STEP 1 UNTIL N DO HDM1F0000 165
[J]:=YMID1[J]-FH[INDX,J]×T1 ;CYI :=INDX +TQP1 ;I3 :=CYI -QP1 ;FOR K :=1 0000 166
STEP 1 UNTIL M DO BEGIN I1 :=(CYI -K)MOD TQP1 ;I2 :=(I3 +K)MOD TQP1 ;T1 0000 167
:=MCOEFF[K]-H ;T2 :=MCOEFF[QP1 -K];FOR J :=1 STEP 1 UNTIL N DO HDM1F[J]:0000 168
=HDM1F[J]-FH[I1,J]×T1 -FH[I2,J]×T2 END ;FOR J :=1 STEP 1 UNTIL N DO HDM10000 169
FMID[J]:=HDM1F[J];DFLAG :=FALSE ;ACCEPT:FOR I :=1 STEP 1 UNTIL M DO BEGI0000 170
N CYI :=(INDX :=(INDX +1)MOD TQP1)+TQP1 ;X :=XF -(C2 -I)×H ;I1 :=(CYI -10000 171
)MOD TQP1 ;T1 :=PCOEFF[0];T2 :=CCOEFF[1];FOR J :=1 STEP 1 UNTIL N DO BEG0000 172
IN YP[J]:=(T4 :=(HDM1F[J]:=HDM1F[J]+H ×(T3 :=FH[I1,J])))+T1 ×T3 ;CS[J]:=0000 173
T4 +T2 ×T3 END ;I3 :=CYI -QP1 ;FOR K :=2 STEP 1 UNTIL M DO BEGIN I1 :=(C0000 174
YI -K)MOD TQP1 ;I2 :=(I3 +K)MOD TQP1 ;T1 :=PCOEFF[K -1];T2 :=CCOEFF[K];T0000 175
3 :=PCOEFF[Q -K];T4 :=CCOEFF[QP1 -K];FOR J :=1 STEP 1 UNTIL N DO BEGIN Y0000 176
P[J]:=YP[J]+T1 ×(T5 :=FH[I1,J])+T3 ×(T6 :=FH[I2,J]);CS[J]:=CS[J]+T2 ×T5 0000 177
+T4 ×T6 END END ;I1 :=(CYI -Q)MOD TQP1 ;I2 :=(CYI -QP1)MOD TQP1 ;T1 :=PC0000 178
OEFF[Q -1];T2 :=CCOEFF[Q];T3 :=PCOEFF[Q];FOR J :=1 STEP 1 UNTIL N DO BEG0000 179
IN YP[J]:=YP[J]+T1 ×(T4 :=FH[I1,J])+T3 ×FH[I2,J];CS[J]:=CS[J]+T2 ×T4 END0000 180
 ;T2 :=CCOEFF[0];CORRECTCNT :=1 ;CORRECT:F(N,X,YP,FP);FOR J :=1 STEP 1 U0000 181
NTIL N DO IF (T1 :=ABS((T3 :=(YC[J]:=CS[J]+T2 ×FP[J]))-YP[J]))>EAV[J]THE0000 182
N BEGIN IF T1 >ERV[J]×ABS(T3)THEN BEGIN FOR J :=J +1 STEP 1 UNTIL N DO Y0000 183
C[J]:=CS[J]+T2 ×FP[J];F(N,X,YC,FP);FOR J :=1 STEP 1 UNTIL N DO IF (T1 :=0000 184
ABS((T3 :=(YP[J]:=CS[J]+T2 ×FP[J]))-YC[J]))>EAV[J]THEN BEGIN IF T1 >ERV[0000 185
J]×ABS(T3)THEN BEGIN FOR J :=J +1 STEP 1 UNTIL N DO YP[J]:=CS[J]+T2 ×FP[0000 186
J];CORRECTCNT :=CORRECTCNT +2 ;IF CORRECTCNT >8 THEN BEGIN INDX :=(CYI -0000 187
I)MOD TQP1 ;GO TO L5 END ;GO TO CORRECT END END ;F(N,X,YP,FH[INDX,*]);PF0000 188
LAG :=TRUE ;CORRECTCNT :=CORRECTCNT +1 ;GO TO L2 END END ;F(N,X,YC,FH[IN0000 189
DX,*]);PFLAG :=FALSE ;L2:END ;I1 :=(CYI -M)MOD TQP1 ;T1 :=MCOEFF[M];FOR 0000 190
J :=1 STEP 1 UNTIL N DO YM[J]:=HDM1FMID[J]+T1 ×FH[I1,J];I3 :=CYI -Q ;FOR0000 191
 K :=0 STEP 1 UNTIL MM1 DO BEGIN I1 :=(CYI -K)MOD TQP1 ;I2 :=(I3 +K)MOD 0000 192
TQP1 ;T1 :=MCOEFF[K];T2 :=MCOEFF[Q -K];FOR J :=1 STEP 1 UNTIL N DO YM[J]0000 193
```

```
:=YM[J]+T1 ×FH[I1,J]+T2 ×FH[I2,J]END ;IF DFLAG THEN BEGIN FOR J :=1 STEP
 1 UNTIL N DO IF (T2 :=ABS((T3 :=Y[J])−YM[J]))>EAVD[J]THEN BEGIN IF T2 >
ERVD[J]×ABS(T3)THEN BEGIN IF T2 >EAV[J]THEN BEGIN IF T2 >ERV[J]×ABS(T3)T
HEN GO TO L4 END ;GO TO L3 END END ;C2 :=C2 −M ;C2 :=C2 /2.0 ;IF PFLAG T
HEN FOR J :=1 STEP 1 UNTIL N DO Y[J]:=YP[J]ELSE FOR J :=1 STEP 1 UNTIL N
 DO Y[J]:=YC[J];C1 :=C1 DIV 2 ;IF C2 <M THEN GO TO CLOSER ;INDX :=INDX +
1 ;FOR K :=1 STEP 1 UNTIL Q DO BEGIN INDX :=(INDX +1)MOD TQP1 ;I1 :=(IND
X +K)MOD TQP1 ;FOR J :=1 STEP 1 UNTIL N DO FH[INDX,J]:=FH[I1,J]END ;GO T
O DOUBLER END ;J :=0 ;L3:FOR J :=J +1 STEP 1 UNTIL N DO IF (T2 :=ABS((T3
 :=Y[J])−YM[J]))>EAV[J]THEN BEGIN IF T2 >ERV[J]×ABS(T3)THEN BEGIN L4:IND
X :=(CYI −M)MOD TQP1 ;L5:C1 :=C1 +C1 ;X :=XF −C2 ×H ;C2 :=C2 +C2 ;GO TO 
STARTER END END ;C2 :=C2 −M ;IF C2 ≥M THEN BEGIN IF PFLAG THEN FOR J :=1
 STEP 1 UNTIL N DO BEGIN YMID1[J]:=Y[J];Y[J]:=YP[J];HDM1FMID[J]:=HDM1F[J
]END ELSE FOR J :=1 STEP 1 UNTIL N DO BEGIN YMID1[J]:=Y[J];Y[J]:=YC[J];H
DM1FMID[J]:=HDM1F[J]END ;DFLAG :=TRUE ;GO TO ACCEPT END ;IF PFLAG THEN F
OR J :=1 STEP 1 UNTIL N DO Y[J]:=YP[J]ELSE FOR J :=1 STEP 1 UNTIL N DO Y
[J]:=YC[J];CLOSER:IF C2 >0 THEN SHANKS(N,X,XF,Y,F,RKSFN,RKSORDER,RKSCOEF
F,P,EA,ER,ABS(INT)/C1);END ;PROCEDURE SHANKS(N,XI,XF,YV,F,M,ORDER,CF,P,E
A,ER,DX);VALUE N,XI,XF,M,ORDER,P,DX;INTEGER N,M,ORDER;REAL XI,XF,P,DX;RE
AL ARRAY YV,CF,EA,ER[0];PROCEDURE F;BEGIN INTEGER I,J,K,L,COUNT,COUNT2,I
I,NCF;INTEGER NCF1;INTEGER DKTR;REAL EFACT;REAL BETA,DCOUNT,DXD,DXH,DXT,
EFACTOR,ERANGE,ES,GAMMA,X,XM;BOOLEAN CFSW,DSW;REAL ARRAY CFD[0:(M+3)×M−2
],FV[0:M−1,0:N],GV,YC,YM,YP[0:N];DEFINE CFH=CFD#;LABEL L1,L2,EXIT;INTEGE
R STEPR,STEPS;M←M−1;STEPS←STEPR←0;DXD←DXT←XF−XI;IF DXT=0THEN GO TO EXIT;
IF DX=0THEN DX←DXD;COUNT←1;WHILE ABS(DX)<ABS(DXD)DO BEGIN COUNT←COUNT+CO
UNT;DXD←DXT/COUNT;END;COUNT2←COUNT+COUNT;DXH←DXT/COUNT2;DCOUNT←COUNT;EFA
CT←1;FOR I←1STEP 1UNTIL ORDER DO EFACT←EFACT+EFACT;ERANGE←0.125/EFACT;EF
ACT←4/EFACT;EFACTOR←COUNT*P×EFACT;DKTR←0;NCF1←(M×M+M)DIV 2+M+M;NCF←NCF1+
1;CFSW←FALSE;FOR I←0STEP 1UNTIL NCF1 DO BEGIN CFD[I]←CF[I]×DXD;CFH[I+NCF
]←CF[I]×DXH;END;X←XI;XM←XI+DXH;L1:DSW←TRUE;F(N,X,YV,GV);IF CFSW THEN L←N
CF1 ELSE L←−1;FOR I←1STEP 1UNTIL M DO BEGIN II←I−1;L←L+1;BETA←CFD[L];FOR
 K←1STEP 1UNTIL N DO YP[K]←GV[K]×BETA+YV[K];FOR J←1STEP 1UNTIL II DO BEG
IN L←L+1;BETA←CFD[L];FOR K←1STEP 1UNTIL N DO YP[K]←FV[J,K]×BETA+YP[K];EN
D;L←L+1;F(N,CFD[L]+X,YP,FV[I,*]);END;L←L+1;GAMMA←CFD[L];FOR K←1STEP 1UNT
IL N DO YP[K]←GV[K]×GAMMA+YV[K];FOR I←1STEP 1UNTIL M DO BEGIN L←L+1;GAMM
A←CFD[L];FOR K←1STEP 1UNTIL N DO YP[K]←FV[I,K]×GAMMA+YP[K];END;L2:IF CFS
W THEN L←−1;FOR I←1STEP 1UNTIL M DO BEGIN II←I−1;L←L+1;BETA←CFH[L];FOR K
←1STEP 1UNTIL N DO YM[K]←GV[K]×BETA+YV[K];FOR J←1STEP 1UNTIL II DO BEGIN
 L←L+1;BETA←CFH[L];FOR K←1STEP 1UNTIL N DO YM[K]←FV[J,K]×BETA+YM[K];END;
L←L+1;F(N,CFH[L]+X,YM,FV[I,*]);END;L←L+1;GAMMA←CFH[L];FOR K←1STEP 1UNTIL
```

```
 N DO YM[K]←GV[K]×GAMMA+YV[K];FOR I←1STEP 1UNTIL M DO BEGIN L←L+1;GAMMA←0000 234
CFH[L];FOR K←1STEP 1UNTIL N DO YM[K]←FV[I,K]×GAMMA+YM[K];END;F(N,XM,YM,F0000 235
V[0,*]);IF CFSW THEN L←-1ELSE L←NCF1;FOR I←1STEP 1UNTIL M DO BEGIN II←I-0000 236
1;FOR K←1STEP 1UNTIL N DO YC[K]←YM[K];FOR J←0STEP 1UNTIL II DO BEGIN L←L0000 237
+1;BETA←CFH[L];FOR K←1STEP 1UNTIL N DO YC[K]←FV[J,K]×BETA+YC[K];END;L←L+0000 238
1;F(N,CFH[L]+XM,YC,FV[I,*]);END;FOR K←1STEP 1UNTIL N DO YC[K]←YM[K];FOR 0000 239
I←0STEP 1UNTIL M DO BEGIN L←L+1;GAMMA←CFH[L];FOR K←1STEP 1UNTIL N DO YC[0000 240
K]←FV[I,K]×GAMMA+YC[K];END;FOR K←1STEP 1UNTIL N DO BEGIN ES←ABS(YC[K]-YP0000 241
[K])×EFACTOR;IF ES≠0THEN BEGIN IF ES≥EA[K]THEN IF ES≥ABS(YC[K])×ER[K]THE0000 242
N BEGIN DSW←FALSE;STEPR←STEPR+1;COUNT←COUNT2;COUNT2←COUNT+COUNT;DCOUNT←D0000 243
COUNT+DCOUNT;DXD←DXH;DXH←DXT/COUNT2;EFACTOR←COUNT*P×EFACT;IF CFSW THEN B0000 244
EGIN FOR I←0STEP 1UNTIL NCF1 DO CFH[I+NCF]←CF[I]×DXH;CFSW←FALSE;END ELSE0000 245
 BEGIN FOR I←0STEP 1UNTIL NCF1 DO CFH[I]←CF[I]×DXH;CFSW←TRUE;END;XM←(COU0000 246
NT2+1-DCOUNT-DCOUNT)×DXH+XI;FOR K←1STEP 1UNTIL N DO YP[K]←YM[K];GO TO L20000 247
;END;IF DSW THEN IF ES≥EA[K]×ERANGE THEN IF ES≥ABS(YC[K])×ER[K]×ERANGE T0000 248
HEN DSW←FALSE;END;END;DCOUNT←DCOUNT-1;STEPS←STEPS+1;FOR K←1STEP 1UNTIL N0000 249
 DO YV[K]←YC[K];IF DCOUNT=0THEN GO TO EXIT;X←(COUNT-DCOUNT)×DXD+XI;IF DC0000 250
OUNT<2THEN BEGIN IF DCOUNT=1THEN DSW←FALSE;IF DCOUNT<1OR DSW THEN BEGIN 0000 251
IF DCOUNT>1THEN DKTR←DKTR+1;COUNT←DCOUNT+1;COUNT2←2;EFACTOR←EFACT;XI←X;D0000 252
XD←DXT←XF-XI;DXH←DXD/2;XM←XI+DXH;CFSW←FALSE;FOR I←0STEP 1UNTIL NCF1 DO B0000 253
EGIN CFD[I]←CF[I]×DXD;CFH[I+NCF]←CF[I]×DXH;END;GO TO L1;END;END;IF DSW T0000 254
HEN BEGIN DKTR←DKTR+1;COUNT2←COUNT;COUNT←COUNT DIV 2;DCOUNT←DCOUNT/2;DXH0000 255
←DXD;DXD←DXT/COUNT;EFACTOR←COUNT*P×EFACT;IF CFSW THEN BEGIN FOR I←0STEP 0000 256
1UNTIL NCF1 DO CFD[I]←CF[I]×DXD;CFSW←FALSE;END ELSE BEGIN FOR I←0STEP 1U0000 257
NTIL NCF1 DO CFD[I+NCF]←CF[I]×DXD;CFSW←TRUE;END;END;XM←(COUNT2+1-DCOUNT-0000 258
DCOUNT)×DXH+XI;GO TO L1;EXIT:END;
    MONITOR INTOVR;                                                         00012000
    FORMAT FMINTOVR("   INTEGER OVERFLOW");                                 00013000
    LABEL L39;                                                              00014000
    FORMAT F2I1(2I1);                                                       00015000
    FORMAT F4E1I(4E20.9,I9);                                                00016000
    REAL TEMP;                                                              00017000
    DEFINE AC =ALLJ IF (TEMP :=ABS(Y [J]-Y1[J]))>EA[J]×2 AND TEMP >ER[J]00018000
    ×ABS(Y [J]+Y1[J])THEN WRITE(PUN ,F4E1I,A,C,Y1[J],Y [J],J)#;          00019000
    DEFINE SS =START,SHANKS#;                                               00020000
    FORMAT FM5I10 (4(2I1,I8,X2),2(I1,I8,X2),I10);                           00021000
    REAL A,C,DDX;                                                           00022000
    OWN ARRAY COEF[0:3,0:3,0:39];                                           00023000
    OWN INTEGER ARRAY Q[-2:73];                                             00024000
    OWN INTEGER KX,KY,KX33,KX41,KY33,KY41;                                  00025000
OWN INTEGER J,S,K;
OWN INTEGER ARRAY AT,BJ[0:3];
LIST LST(FOR J:=-2 STEP 1 UNTIL 72 DO Q[J]);
DEFINE HEREBEFORE =Q[73]#,ALLJ =FOR J :=1 STEP 1 UNTIL N DO #,YOBYY
=ALLJ YO[J]:=Y[J]#,SETY1ANDY =ALLJ
BEGIN
  Y1[J]:=Y[J];
  Y[J]:=YO[J]
END #;
LABEL L29,L1,L2,L3,L4,L5,L6,L7,M00,M01,M02,M03,M10,M11,M12,M13,M20,
M21,M22,M23,M30,M31,M32,M33;
SWITCH SW :=M00,M01,M02,M03,M10,M11,M12,M13,M20,M21,M22,M23,M30,M31
,M32,M33;
SWITCH RETURN :=L1,L2,L3,L4,L5,L6,L7;
INTEGER L,ALF,ALF1,ALF2,BET,BET1,BET2,ALFBEST,BESTO,TALF,TALF1,TALF2
,TBET,TBET1,TBET2,BETBEST,BESTM ;
ARRAY YO,Y1[0:N];
PROCEDURE RECORDORDERS(M,X,XT,Y,YT);
VALUE M,X,XT,Y,YT;
INTEGER M,X,XT,Y,YT;

BEGIN
  KX :=KY :=1 +8×M;
  KX :=KX +2×X;
  KY :=KY +2×Y;
  IF XT <YT THEN
  BEGIN
    Q[KY+1]:=Q[KY+1]-YT;
    Q[KX ]:=Q[KX ]+XT;

  END;
  IF YT <XT THEN
  BEGIN
    Q[KX+1]:=Q[KX+1]-XT;
    Q[KY ]:=Q[KY ]+YT;

  END;

END;
PROCEDURE RECORDMETHODS(X,XT,Y,YT);
```

```
VALUE X,XT,Y,YT;
INTEGER X,XT,Y,YT;

BEGIN
  KX33 :=33 +2×X;
  KY33 :=33 +2×Y;
  KX41 :=8 +KX33 +8×Y;
  KY41 :=8 +KY33 +8×X;
  IF XT <YT THEN
  BEGIN
    Q[KY33+1]:=Q[KY33+1]-YT;
    Q[KX33 ]:=Q[KX33 ]+XT;
    Q[KY41+1]:=Q[KY41+1]-YT;
    Q[KX41 ]:=Q[KX41 ]+XT;

  END;
  IF YT <XT THEN
  BEGIN
    Q[KX33+1]:=Q[KX33+1]-XT;
    Q[KY33 ]:=Q[KY33 ]+YT;
    Q[KX41+1]:=Q[KX41+1]-XT;
    Q[KY41 ]:=Q[KY41 ]+YT;

  END;

END;
PROCEDURE SELECTORDERS(M,X,Y);
INTEGER M,X,Y;

BEGIN
  J:=Q[0]:=(Q[0]×4093 +3000001)MOD 16777216 ;
  J :=J/@4;
  FOR S :=0 STEP 1 UNTIL 3 DO
  BEGIN
    J :=(J+1)MOD 4;
    K :=1 +8×M +2×J;
    AT[S]:=Q[K]+Q[K+1];
    BJ[S]:=J;

  END;
```

```
      X:=BJ[0];
      Y :=IF AT[1]≥AT[2]THEN IF AT[1]≥AT[3]THEN BJ[1]ELSE BJ[3]ELSE IF
      AT[2]≥AT[3]THEN BJ[2]ELSE BJ[3];

    END;
    PROCEDURE SELECTMETHODS(X,Y);
    INTEGER X,Y;

    BEGIN
      J:=Q[0]:=(Q[0]×4093 +3000001)MOD 16777216 ;
      J :=J/@4;
      FOR S :=0 STEP 1 UNTIL 3 DO
      BEGIN
        J :=(J+1)MOD 4;
        K :=33 +2×J;
        AT[S]:=Q[K]+Q[K+1];
        BJ[S]:=J;

      END;
      X:=BJ[0];
      Y :=IF AT[1]≥AT[2]THEN IF AT[1]≥AT[3]THEN BJ[1]ELSE BJ[3]ELSE IF
      AT[2]≥AT[3]THEN BJ[2]ELSE BJ[3];

    END;
    IF HEREBEFORE ≠"YES"THEN
    BEGIN
      LABEL L1;
      FILE IN TAPE831 2(2,90);
      FORMAT FM ("SEND THIS HISTORY FILE AND THE PUNCHED CARDS TO L J GA
LLAHER VIA ","CAMPUS MAIL"////X5,A5,I10,I10/(8I10));
      FORMAT FORM(/(5E20.11));
      INTEGER J,K,S;
      WHILE TRUE DO READ(A831HST,75,Q[*])[L1:L1];
      L1:REWIND(A831HST);
      COMMENT READ COEF.FILE;
      FOR J :=0 STEP 1 UNTIL 3 DO FOR K :=0 STEP 1 UNTIL 3 DO
      BEGIN
        IF J=2 AND K=0 THEN SPACE(TAPE831,1);
        IF J=1 AND K=0 THEN SPACE(TAPE831,5);
        READ(TAPE831,40,COEF[J,K,*]);
      END;
      WRITE(PUN ,FM,LST);
      HEREBEFORE :="YES";

    END;
    INTOVR :=L39;
    SELECTMETHODS(ALF,BET);
    SELECTORDERS(ALF,ALF1,ALF2);
    SELECTORDERS(BET,BET1,BET2);
    C :=XI;
    DDX:=(XF-XI)/8;
    A:=C;
    YOBYY;
    C:=A+DDX;
    WRITE(PUN ,F2I1,ALF,ALF1);
    TALF1 :=TIME(2);
    L:=1;
    GO TO SW[4×ALF+ALF1+1];
    L1:TALF1 :=TIME(2)-TALF1;
    SETY1ANDY;
    WRITE(PUN ,F2I1,ALF,ALF2);
    TALF2 :=TIME(2);
    L:=2;
    GO TO SW[4×ALF+ALF2+1];
    L2:TALF2 :=TIME(2)-TALF2;
    AC;
    A:=C;
    YOBYY :=(Y[J]+Y1[J])/2;
    C:=A+DDX;
    WRITE(PUN ,F2I1,BET,BET1);
    TBET1 :=TIME(2);
    L:=3;
    GO TO SW[4×BET+BET1+1];
    L3:TBET1 :=TIME(2)-TBET1;
    SETY1ANDY;
    WRITE(PUN ,F2I1,BET,BET2);
    TBET2 :=TIME(2);
    L:=4;
    GO TO SW[4×BET+BET2+1];
```

```
L4:TBET2 :=TIME(2)-TBET2;
AC;
A:=C;
YOBYY :=(Y[J]+Y1[J])/2;
C:=A+DDX×2;
ALFBEST :=IF TALF1≤TALF2 THEN ALF1 ELSE ALF2;
BETBEST :=IF TBET1≤TBET2 THEN BET1 ELSE BET2;
WRITE(PUN ,F2I1,ALF,ALFBEST);
TALF :=TIME(2);
L:=5;
GO TO SW[4×ALF+ALFBEST+1];
L5:TALF :=TIME(2)-TALF ;
SETY1ANDY;
WRITE(PUN ,F2I1,BET,BETBEST);
TBET :=TIME(2);
L:=6;
GO TO SW[4×BET+BETBEST+1];
L6:TBET :=TIME(2)-TBET ;
AC;
A:=C;
YOBYY :=(Y[J]+Y1[J])/2;
C:=XF;
FOR J :=1 STEP 1 UNTIL 40 DO Q[J]:=Q[J]×0.980;
RECORDORDERS(ALF,ALF1,TALF1,ALF2,TALF2);
RECORDORDERS(BET,BET1,TBET1,BET2,TBET2);
RECORDMETHODS(ALF,TALF,BET,TBET);
Q[-1]:=Q[-1]+1;
WRITE(A831HST,75,Q[*]);
WRITE(PUN ,FM5I10,ALF,ALF1,TALF1,ALF,ALF2,TALF2,BET,BET1,TBET1,BET,
BET2,TBET2,ALF,TALF,BET,TBET,Q[-1]);
IF TALF ≤TBET THEN
BEGIN
  BESTM :=ALF;
  BESTO :=ALFBEST
END ELSE
BEGIN
  BESTM :=BET;
  BESTO :=BETBEST
END;
L :=7;
```

```
GO TO SW[4×BESTM +BESTU +1];

BEGIN
  GO TO L29;
  M00:COMMENT;
  ADAMS(N,A,C,Y,F,P,3,DX,EA,ER,COEF[0,0,*],4,4,COEF[3,0,*],SS );
  GO TO L29;
  M01:COMMENT;
  ADAMS(N,A,C,Y,F,P,4,DX,EA,ER,COEF[0,1,*],4,4,COEF[3,0,*],SS );
  GO TO L29;
  M02:COMMENT;
  ADAMS(N,A,C,Y,F,P,5,DX,EA,ER,COEF[0,2,*],4,4,COEF[3,0,*],SS );
  GO TO L29;
  M03:COMMENT;
  ADAMS(N,A,C,Y,F,P,6,DX,EA,ER,COEF[0,3,*],5,5,COEF[3,1,*],SS );
  GO TO L29;
  M10:COMMENT;
  BUTCHER(N,A,C,2,EA,ER,DX,COEF[1,0,*],F,P,COEF[3,0,*],SS,Y,4,4);
  GO TO L29;
  M11:COMMENT;
  BUTCHER(N,A,C,3,EA,ER,DX,COEF[1,1,*],F,P,COEF[3,0,*],SS,Y,4,4);
  GO TO L29;
  M12:COMMENT;
  BUTCHER(N,A,C,4,EA,ER,DX,COEF[1,2,*],F,P,COEF[3,0,*],SS,Y,4,4);
  GO TO L29;
  M13:COMMENT;
  BUTCHER(N,A,C,4,EA,ER,DX,COEF[1,2,*],F,P,COEF[3,1,*],SS,Y,5,5);
  GO TO L29;
  M20:COMMENT;
  COWELL(N,A,C,Y,F,EA,ER,P,DX,5,5,COEF[3,1,*],4,COEF[2,0,*],SS);
  GO TO L29;
  M21:COMMENT;
  COWELL(N,A,C,Y,F,EA,ER,P,DX,5,5,COEF[3,1,*],6,COEF[2,1,*],SS);
  GO TO L29;
  M22:COMMENT;
  COWELL(N,A,C,Y,F,EA,ER,P,DX,5,5,COEF[3,1,*],8,COEF[2,2,*],SS);
  GO TO L29;
  M23:COMMENT;
  COWELL(N,A,C,Y,F,EA,ER,P,DX,5,5,COEF[3,1,*],10,COEF[2,3,*],SS);
  GO TO L29;
```

```
      M30:COMMENT;
      SHANKS(N,A,C,Y,F,4,4,COEF[3,0,*],P,EA,ER,DX);
      GO TO L29;
      M31:COMMENT;
      SHANKS(N,A,C,Y,F,5,5,COEF[3,1,*],P,EA,ER,DX);
      GO TO L29;
      M32:COMMENT;
      SHANKS(N,A,C,Y,F,6,6,COEF[3,2,*],P,EA,ER,DX);
      GO TO L29;
      M33:COMMENT;
      SHANKS(N,A,C,Y,F,7,7,COEF[3,3,*],P,EA,ER,DX);
      L29:GO TO RETURN[L];
      L39:WRITE(PUN,FMINTOVR);
      GO TO M30;

    END;
    L7:
  END;
```

IV. RESULTS AND CONCLUSIONS

A. Applications

Three types of problems were used to exercise this integration procedure. The first type is the Arenstorf orbits of the restricted three body problem. The second is the system of linear differential equations associated with Fourier transforms. The third type is the system of linear equations obtained from a discretization of the partial differential equation for the vibrating string.

The first of these is characterized by the necessity of frequent step size change. The other two types are characterized by having a large number (20 to 100) of coupled equations.

B. Results

The executive routine performed quite satisfactorily. Learning took place as was desired, the procedure adapting readily to the characteristics of a particular problem and accuracy.

The results of running with a variety of problems and accuracies are that no particular method seems to be exceptionally superior to any other. It did appear that for the accuracy range used (10^{-3} to 10^{-9}) certain orders of some methods were inappropriate. Also for a given method one particular order usually dominated, but which one dominated depended on the accuracy being asked and to some extent on the problem.

All methods performed well and, for different problems, different methods showed up more successfully. The Runge-Kutta-Shanks method was usually faster for problems where frequent step size changes were required, but the multi-step methods usually performed better when long runs of uniform step size were appropriate. Of the multistep methods, that of Adams was usually the fastest.

The performance of the various orders of each method was as follows:

For the Adams method, 6th order was best most often for these accuracies, with 5th order next fastest.

Of the Butcher formulas, the 5th order was most often the fastest. No clear cut case was established for second best, but it was evident that 9th order or higher was clearly too slow at these accuracies to be included among the possible orders.

For the Cowell method, 6th order was usually the best. 12th order and higher were too slow and should not be used at these accuracies.

Of the Shanks formulas, the 4th order was usually the fastest, with the 5th and 6th orders not too far behind.

C. Conclusions

The results justify the conclusion that the present program would be suitable and effective as a general library program for integrating systems of differential equations. It was evident that no particular method or order is exceptionally superior to all the others. Depending on the accuracy and the problem, different methods and orders are best. The executive routine does a satisfactory job of finding a good method and order for each individual problem.

D. Suggestions for Further Study

Several additional tasks and improvements to the present project can be envisioned.

The first additional task would be to convert the integration procedure to double precision (22 decimal places). This would allow an exploration of a wider range of accuracies and order. Also at the higher accuracies more striking differences in the efficiencies of the various methods and orders are expected to occur. Experiments of the type carried out in single precision could then be done in double precision and the results extended over a wider range of accuracies and orders.

As a second additional task, a further investigation should be carried out into the correlation between order and accuracies. The present program does not try to anticipate the optimum order from the accuracy requirements. There should be a correlation between accuracy and optimum order. This could be built into the program either on an empirical basis or preferably as a learning function; that is, as a correlation to be learned by the program from the running experience.

A third suggestion for further work would be to make improvements in the learning mechanism. One such possibility just mentioned is to incorporate the learning of the correlation between order and accuracy. Also an investigation of the optimum rate of "forgetting" could be undertaken. The whole mechanism of learning should be investigated more thoroughly for the purpose of optimizing the learning process.

Other revisions in the program or extensions of this work would be to improve or refine the step size and error control, to do more experimenting with a wider variety of problems, and possibly to incorporate other integration methods into the program.

Respectfully submitted,

I. E. Perlin
Project Director

V. BIBLIOGRAPHY

References on the Adams Method

1. Krogh, Fred T., J. ACM, 13, (1966) 374-385.

2. Bashforth, F., and Adams, J. C., Theories of Capillary Action, Cambridge University Press, 1883.

3. Dahlquist, Germund, Math. Scan., 4, (1956) 33.

4. Dahlquist, Germund, Stockholm Tekniska Hogskolan, No. 130, (1959), 1-87.

5. Henrici, Peter, Discrete Variable Methods in Ordinary Differential Equations, John Wiley and Sons, New York, 1962, 191-199, 200, 203-204, 224, 241, 258, 272-273, 274.

6. Newberry, A. C. R., Math. Comp. 17, 84, (1963) 452-455.

References on Cowell (Constant Nth Order Difference) Method

7. Cowell, P. H., and Crommelin, A. C. D., "Investigation of the Motion of Halley's Comet from 1759 to 1910." Appendix to Greenwich Observations for 1909, Edinburgh, 1910, 84.

8. Herrick, S., "Step-by-Step Integration of $\dot{x} = f(x, y, z, t)$ without a Corrector," Mathematical Tables and Other Aids to Computation, No. 34, April 1951, 61-67.

9. Durham, H. L., Jr., et al, "Study of Methods for Numerical Solution of Ordinary Differential Equations," Final Report, Contract NAS8-11129, Project A-740, Engineering Experiment Station, Georgia Institute of Technology, Atlanta, Georgia, 1964.

10. Currie, J. C., et al, "Study of Satellite Orbit Computation Methods and An Error Analysis of the Mathematical Procedures," Technical Report No. 3, Prepared for Department of the Air Force Contract No. AF29(600)-1756, Project No. 1770, Headquarters Air Force Missile Development Center, Air Research and Development Command, Holloman Air Force Base, New Mexico, August 1959.

References on the Stetter-Gragg-Butcher Method

11. Gear, C. W., "Hybrid Methods for Initial Value Problems in Ordinary Differential Equations," J. SIAM (B) 2 (1965) p. 69.

12. Butcher, J. C., "A Modified Multistep Method for the Numerical Integration of Ordinary Differential Equations," J. ACM, 12 (1965) 124-135.

13. Gragg, W. B. and Stetter, H. J., "Generalized Multistep Predictor-Corrector Methods," J. ACM, 11 (1964) 188-209.

References on Runge-Kutta-Fehlberg Method

14. Fehlberg, E., "Runge-Kutta Type Formulas of High-Order Accuracy and their Application to the Numerical Integration of the Restricted Problem of Three Bodies," Presented at the Colloque International des Techniques de Calcul Analogique et Numerique in Aeronautique in Liege, Belgium, September 1963.

15. Fehlberg, E., "New High-Order Runge-Kutta Formulas with Step Size Control for Systems of First- and Second-Order Differential Equations," Presented by S. Filippi at the Meeting of the GAMM in Giessen, Germany, April 1964.

16. Fehlberg, E., "Runge-Kutta Type Formulas of High-Order Accuracy and Their Application to the Numerical Integration of the Restricted Problem of Three Bodies," (Private Communication).

References on the Runge-Kutta-Shanks Method

17. Shanks, E. B., "Formulas for Obtaining Solutions of Differential Equations by Evaluations of Functions," Presented at the summer meeting of the American Mathematical Society in Boulder, Colorado, August 1963, and Private Communication.

Other References

18. Francis, O. B., Jr., et al, "Study of the Methods for the Numerical Solution of Ordinary Differential Equations," Final Report, Project A-831, Contract NAS8-20014, Engineering Experiment Station, Georgia Institute of Technology, Atlanta, Georgia, 1966.